SYSTEM ANALYSIS

For The

Huntsville Operation Support Center
Distributed Computer System

Interim Report for Period March 1985 through June 1985

ANNUAL REPORT

1985

Submitted by:

D. Massey, Associate Investigator
F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:

NASA MSFC, Alabama
Technical Monitor: Frank Emmens, EB32
(205) 453-4629

NAS8-34906

SYSTEM ANALYSIS

For The

Huntsville Operation Support Center
Distributed Computer System

Interim Report for Period March 1985 through June 1985

ANNUAL REPORT

1985

Submitted by:

D. Massey, Associate Investigator
F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:

NASA MSFC, Alabama
Technical Monitor: Frank Emmens, EB32
(205) 453-4629

NAS8-34906

OVERVIEW OF REPORT

Network Systems Corporation's HYPERchannel network is a baseband, 50 Megabit per second, high speed local area network. HYPERchannel utilizes a Carrier Sense Multiple Access (CSMA) data trunk access protocol, with prioritized staggered delays. This network has the capability of interconnecting various computer resources of differing manufacture, into a single efficient computing facility.

This network is currently in use at the Huntsville Operations Support Center (HOSC). HOSC as a distributed computing system, is responsible for data acquisition and analysis during National Aeronautics and Space Administration's (NASA) Space Shuttle operations. HOSC also provides computing services for Marshall Space Flight Center's (MFSC) non-mission activities. As mission and non-mission activities change, so do the support functions of HOSC change, demonstrating the need for some method of simulating activity at HOSC in various configurations

The simulation developed in this work primarily models NSC's HYPERchannel network, since this network basically determines HOSC's efficiency in accomplishing its mission. The model simulates the activity of a steady-state network, reporting statistics such as, transmitted bits, collision statistics, frame sequences transmitted, and average message delay. These statistics may be used to evaluate such performance indicators as throughput, utilization, and delay.Thus the overall performance of the network may be evaluated, as well as predicting possible overload conditions.

HOW TO READ THIS REPORT

This report is a fully detailed report concerning the inner workings of the HYPERchannel Local Area Network, Chapter 2; a description of the HOSC system, Chapter 3; a full detailed description of the simulation software (design architecture Chapter 4, individual software module description Appendix III); and a presentation of the simulation results for single and dual trunk operation, Chapter 5. The reader interested in a quick overview and a survey of the HOSC system performance may read Chapter 1 and Chapter 5 only. However knowledge of the HYPERchannel adapter's inner workings including message frame construction and timing is desirable, Chapter 2.

II, III

TABLE OF CONTENTS

Chapter		Page
	OVERVIEW OF REPORT	iv
	TABLE OF CONTENTS	v
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
	LIST OF ACRONYMS	x
1	Introduction	1
	1.1 NSC's HYPERchannel Local Area Network	1
	1.2 HOSC System Overview	2
	1.3 Research Objective	3
2	HYPERchannel Network Description	6
	2.1 Network Components	6
	2.1.1 Adapter Architecture	9
	2.2 HYPERchannel Protocols	12
	2.2.1 Trunk Access Protocol	13
	2.2.2 Adapter-Adapter Link Level and Virtual Circuit Protocols	19
	2.2.2.1 Frame and Frame Sequence Structure	21
	2.2.2.2 Trunk Selection	30
	2.2.2.3 Virtual Circuit Establishment	32
	2.2.2.4 Data Flow Control	36
	2.3 Intra-Adapter Communication	38
3	HOSC System Description	39
	3.1 Typical HOSC System Activities	40
	3.2 HOSC System Components	45

Chapter Page

4 Software Design Description..... 58
4.1 Software Functions..... 58
4.2 Design Constraints..... 59
4.3 Design Specifications..... 59
4.3.1 Simulation Assumptions..... 59
4.3.2 Parameter Definition..... 60
4.4 Algorithm Description..... 62
4.4.1 Overall Simulation Activity..... 62
4.4.2 Model Set-Up..... 69
4.5 System Files..... 69
4.6 Reported Statistics..... 71
4.7 Validation Criteria..... 72

5 Model Analysis and Conclusions..... 73
5.1 Theoretical Performance Bounds..... 73
5.2 Presentation of Simulation Results.. 81
5.2.1 Single Trunk HOSC Results..... 82
5.2.2 Dual Trunk HOSC Results..... 97
5.3 Analysis and Conclusions.....110

APPENDIX I Software Source Listing.....117

APPENDIX II Summary of ISO-OSI Model of Distributed Networks.....168

APPENDIX III Simulation Software Module Description.....169
1. Sim1.....169
2. Initialize.....169
3. CharacterizeNetwork.....172
4. PrintNetDescription.....172
5. FindNext.....172
6. PickA.....175
7. Acollision.....175
8. Uniform.....176
9. Update.....176
10. Printstats.....179
11. Activitysummary.....181

APPENDIX IV Typical Single Trunk Simulation Results.....183

APPENDIX V Typical Dual Trunk Simulation Results.....190

REFERENCES AND BIBLIOGRAPHY.....197

LIST OF TABLES

Tables Page

2.1 HYPERchannel Protocol Frames 22

2.2 Frame Formats 23

3.1 HOSC Data Transfers 42

3.2 Summary of DEC VAX 11/780 I/O Characteristics 54

3.3 Summary of Perkin Elmer 3240 I/O Characteristics 57

5.1 Theoretical Bounds on a HYPERchannel Trunk of Length 1000 Feet. 77

5.2 Calculation of Parameters using Simulation Statistics 83

5.3 Relative Probabilities of Transmitter/ Receiver Pairs for Figure 5.1 84

5.4 Data Points for Figure 5.2 86

5.5 Data Points for Figure 5.3 88

5.6 Relative Probabilities for Transmitter/ Receiver Pairs for Figure 5.4. 98

5.7 Data Points for Figure 5.5 (Trunk 1) . .100

5.8 Data Points for Figure 5.5 (Trunk 2) . .101

5.9 Data Points for Figures 5.6 and 5.7 . .104

LIST OF FIGURES

Figure		Page
2.1	Typical HYPERchannel Interconnection	7
2.2	Block Diagram of NSC HYPERchannel Adapter	11
2.3	HYPERchannel Delay Time Events	16
2.4	Sample AD8 Contention Timing Calculations	19
2.5	Message-Only Frame Sequence	25
2.6	Timings for Message-Only Sequences	26
2.7	Message-with-Data Frame Sequence	27
2.8	Timings for Message-with Data Sequences	28
2.9	Adapter Reservation Scheme	33
3.1	Typical HOSC Configuration	41
3.2	Block Diagram of VAX 11/780 Computer	47
3.3	Bus Configuration of VAX 11/780	49
3.4	Block Diagram of PE 3244	55
4.1	Message-with-Data Sequence for Simulation	64
4.2	Flow Diagram of Simulation Activity.	65
4.3	Model Set-Up for Simulation.	70
5.1	Single Trunk Simulation Set-Up	85
5.2	Plot of Utilization versus Offered Load for Single Trunk Simulation	87
5.3	Plot of Transmitted Control and Data Bits versus Offered Load for Single Trunk Simulation	90
5.4	Dual Trunk Simulation Set-Up	99

Figure Page

5.5 Plot of Utilization versus Offered Load for Dual Trunk Simulation 102

5.6 Plot of Transmitted Control and Data Bits versus Offered Load for Dual Trunk (Trunk 1) Simulation 105

5.7 Plot of Transmitted Control and Data Bits versus Offered Load for Dual Trunk (Trunk 2) Simulation 106

LIST OF ACRONYMS

ASCII	American Standard Code for Information Interchange
BDP	Buffered Data Path
BR	Bus Request
CATV	Community Area Television
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Acess
DDP	Direct Data Path
DEC	Digital Equipment Corporation
DMA	Direct Memory Access
ECIO	Experiment Computer Input/Output
ET	External Tanks
FEP	Front End Processor
HSLN	High Speed Local Network
HOSC	Huntsville Operation Support Center
I/O	Input/Output
ISO	International Standards Organization
JSC	Johnson Space Center
KSC	Kennedy Space Center
LPS	Launch Processing System
Mbps	Megabits per second
MECO	Main Engine Cut off
MER	Mission Evaluation Room
MIPS	Mission Integration Plannin System
MPS	Main Propulsion System
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
NPR	Non-Processor Request
NSC	Network Systems Corporation
OD	Operational Data
OSI	Open Systems Interconnection
PE	Perkin Elmer
POCC	Payload Operations Control Center
RSS	Range Safety System
SBI	Synchronous Backplane Interface
SRB	Solid Rocket Booster
SSME	Space Shuttle Main Engine
STS	Shuttle Transport System
UBA	UNIBUS Adapter
UBUS	UNIBUS
usec	Microsecond

Chapter 1
Introduction

1.1 NSC's HYPERchannel Local Area Network

Network Systems Corporation's (NSC) HYPERchannel local area network is a widely used very high speed, 50 Megabits per second, digital communications facility. HYPERchannel utilizes microcomputer-based adapters and standard 75 ohm CATV coaxial cable as data trunks, to interconnect various computer resources within a computing installation. There are three types of adapters (Processor, Device, and Link) used to interface individual equipment to the network of data trunks. These adapters provide for:

- interconnection of equipment from various manufacturers.
- high performance communication between processors.
- a sharing of peripheral systems between multiple processors.
- realistic physical location of interconnected equipment.
- increased utilization of interconnected equipment.
- increased tolerance to individual equipment failures.

Finally, Data movement within a HYPERchannel network is accomplished by use of frames and frame sequences. These frames contain the necessary information for addressing and routing, along with associated user messages and data. A sophisticated internal protocol

is used to provide for contention allocation, error detection, retransmission allocation, and data movement. These capabilities provide a reliable network that is insensitive to configuration, and single points of failure. This internal protocol is the basis for HYPERchannel system performance and will be described further in a subsequent chapter.

1.2 HOSC System Overview

The Huntsville Operations Support Center (HOSC), as implemented at the Marshall Space Flight Center (MSC) in Huntsville, Alabama, is a distributed computer resource facility. HOSC is designed to provide real time acquisition, analysis, and display of data during National Aeronautics and Space Administration's (NASA) space missions. HOSC provides support primarily for the Space Shuttle, Space Telescope, and Space Laboratory missions. Among the resources available at HOSC are various large minicomputers and peripherals. A large portion of these resources are interconnected using NSC's HYPERchannel network. The configuration flexibility afforded by HYPERchannel provides HOSC with the ability to adapt to future mission requirements and also provides MSFC with a large computing facility for non-mission activities.

Data handling at HOSC falls into two main categories, mission and non-mission activities. Mission

activities are those that take place during powered flight. For example, the space shuttle main engine data is analyzed by the Main Propulsion System (MPS) processor, which is also sent to a back up processor simultaneously. Data arrives via direct ground links or satellite communication links from the Kennedy Space Center Firing Room at NASA's Kennedy Space Center (KSC) in Florida or from NASA's Johnson Space Center (JSC) in Texas. Services provided by the network during missions are data analysis and presentation services for HOSC mission support teams. Non-mission activities are also provided by HOSC, one of these is support for the Payload Operations Control Center (POCC) preplanning activity. Future expansion at HOSC may include computer resource support for Digital Equipment Corporation's interactive graphics data base (IGDS), and XEROX Corporation's text-processing operation (SIGMA), both of which, are non-mission activities.

1.3 Research Objective

NSC's HYPERchannel local area network is a sophisticated computer network allowing considerable flexibility and adaptability with regard to computer resources. HYPERchannel allows a wide variety of minicomputers, not necessarily manufactured by the same company, to be interconnected in a single or

multi-trunk network. Thus, computer resources which may be separated by distance or manufacturer may be utilized by one another within this distributed computer network. HOSC, as a distributed computing facility, utilizes HYPERchannel to accomplish the interconnection of various computer resources. Mission requirements imposed on the HOSC, require that it possess the ability to adapt and change its resources according to the dictates of a particular mission. The configuration of the HOSC's resources are mission dependent and may be readily changed, however, the effects of those changes on system performance are not readily apparent. The data handling capability of the HYPERchannel network and its performance in various configurations affect HOSC's system performance directly. With this in mind, the primary goal of the research effort was the development of a simulation model of the HYPERchannel network that could be manipulated to predict and characterize the behavior of the HOSC in a variety of scenarios.

The first step in this analysis was an investigation of the operating properties and characteristics of a HYPERchannel network. In Chapter 2 of this document, a functional description f HYPERchannel is detailed with particular emphasis on the protocols used by the network. The second step,

detailed in Chapter 3, was the characterization of various computer resources and possible HOSC activity scenarios. The third step, detailed in Chapter 4, was the definition and development of a software model of a HYPERchannel network. Using this simulation model, system parameters were varied to simulate changes to the operating environment of the HOSC. Finally, the results and conclusions of these manipulations to the simulation model were compiled and analyzed in order to provide some insight into the present and perhaps future operating characteristics of the HOSC.

Chapter 2
HYPERchannel Network Description

2.1 Network Components

Network Systems Corporation's (NSC) HYPERchannel network is a high speed, baseband local area network. A HYPERchannel network consists of standard Community Area Television (CATV) coaxial cables used as data **trunks** and various adapters to control network traffic flow. These adapters use a carrier sense multiple access (CSMA) protocol with prioritized, staggered delays to implement traffic flow and control. Network messages generated by an adapter are composed of **frames** and are called **frame sequences**. These messages are broadcast over a given trunk at a fixed rate of 50 Megabits per second, using a phase modulation technique with a manchester data format, and are received by network components through nondirectional taps or **ports** (Figure 2.1). Network components at these ports are known as **network adapters**, which control access to a given trunk. A network adapter can be attached to as many as four independent network trunks, enabling a configuration to match a specific installation and provide:

- **incremental expansion**
- **linear interconnect cost**
- **no single point of failure**

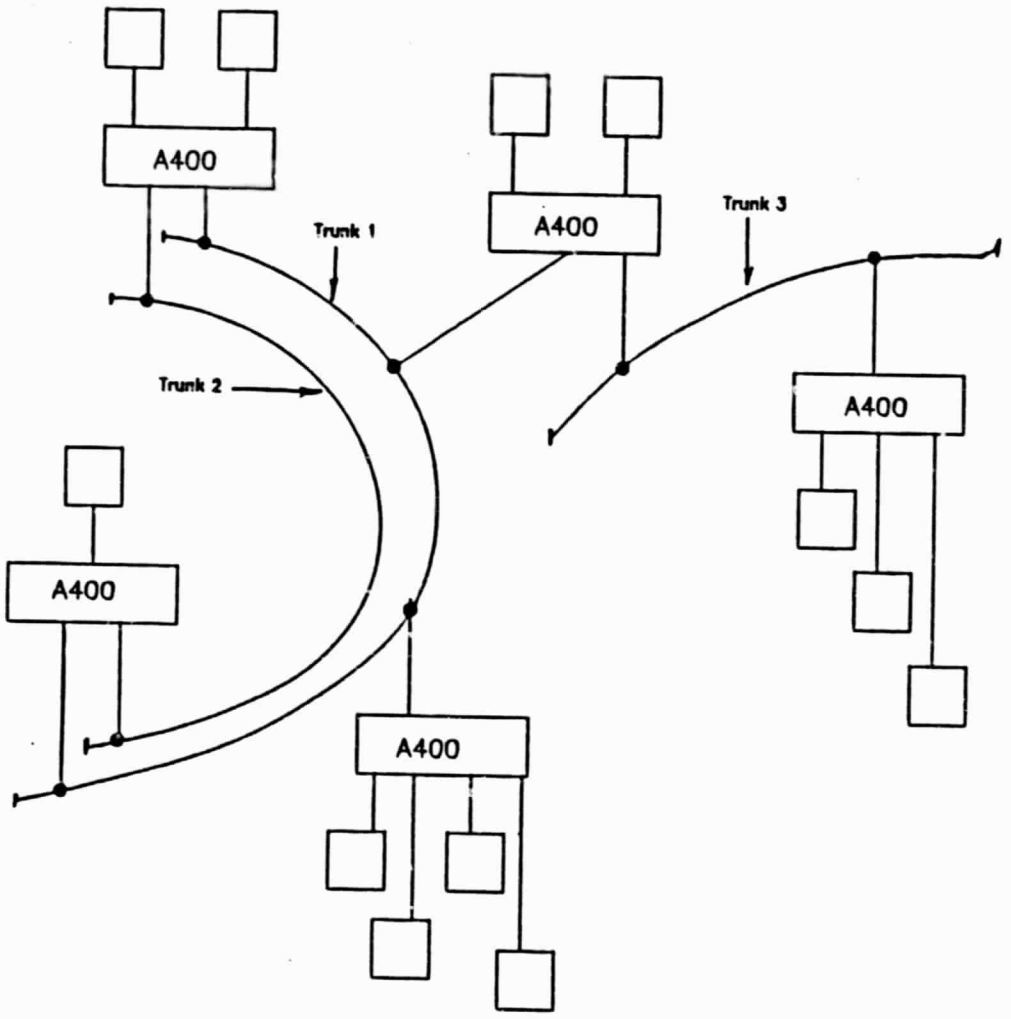


Figure 2.1 Typical HYPERchannel Interconnection.

NSC manufactures three main types of network adapters: processor, device and link. A **processor adapter** interfaces a computer data channel to the network of data trunks. It serves as buffered data path between the computer attached, and the trunk network. The computer communicates with data buffers in the adapter at its own rate and asynchronous with transmissions on the network trunks. The adapter provides the necessary formatting and internal protocol to transmit the contents of its buffers over the selected network trunk. The processor adapter receives data from the trunk network and provides status information to the originator of the data. The computer at the receiving adapter may then access the data and any associated message in the adapter buffer. NSC has designated this adapter type as **A400 HYPERchannel adapters**, herein referred to as adapters. The **device adapter** functions as a remote channel providing a computer data channel for the attachment of peripheral control units. This adapter is driven by network messages from the data trunks. It receives network messages containing device commands, data, and control information. It generates messages containing status or device data. The device adapter operates multiple devices in the same manner as a data channel produced by a computer manufacturer operates multiple devices.

The link adapter is used in pairs to provide communication between remote HYPERchannel networks. A link adapter receiving a network message destined for a remote network transmits this message over a wide band communications link to the remote network. The link adapter at the remote location receives this message and retransmits it over its own local network. There are two types of link adapters, the terrestrial link adapter and the satellite link adapter.

Finally, due to the nature of the HOSC computing facility only the processor adapter will be included in the model.

2.1.1 Adapter Architecture

Each adapter, regardless of the type of user equipment attached, consists of the following:

- A 16-bit micro-processor with 4096 words of read-only memory (ROM)
- Storage section (Buffer and Control logic)
  - 1024 8-bit bytes of control memory with odd parity
  - 4096 8-bit bytes of buffer memory with odd parity
  - 16 working registers
  - 16 trunk registers
  - 256 extension registers
- One trunk interface

This portion of an adapter is termed the **Nucleus Adapter.** The nucleus adapter has two general

applications,(processor and device adapter) determined by the type of equipment attached and the necessary interface required. Thus an adapter consists of a specific device interface and a nucleus adapter. The device interface will then define the particular adapter type. An adapter can be divided into four main sections: micro-processor, buffer, and control logic, trunk interface, and device or equipment interface (Figure 2.2). The **micro-processor** has a 4096 16-bit word read-only instruction memory with a 320 nanosecond cycle time. It handles equipment functions and responses and manages data flow on the direct data path between the equipment and the adapter buffers. The **buffer and control logic** along with the trunk interfaces control the high speed trunk transmissions. It consists of a 1024 8-bit byte control buffer, normally used to stack control information, and a data buffer containing 4096 or (optionally) 8192 8-bit bytes. This section also contains a flag register and other registers to specify buffer address, length of data transmission, and network addressing functions. The control logic contained in this section allow concurrent access to memory by both trunk interfaces and equipment interfaces. The 100 Mbps buffers allow a 50 Mbps data movement on the data trunks along with a 50 Mbps data movement with the attached equipment. The

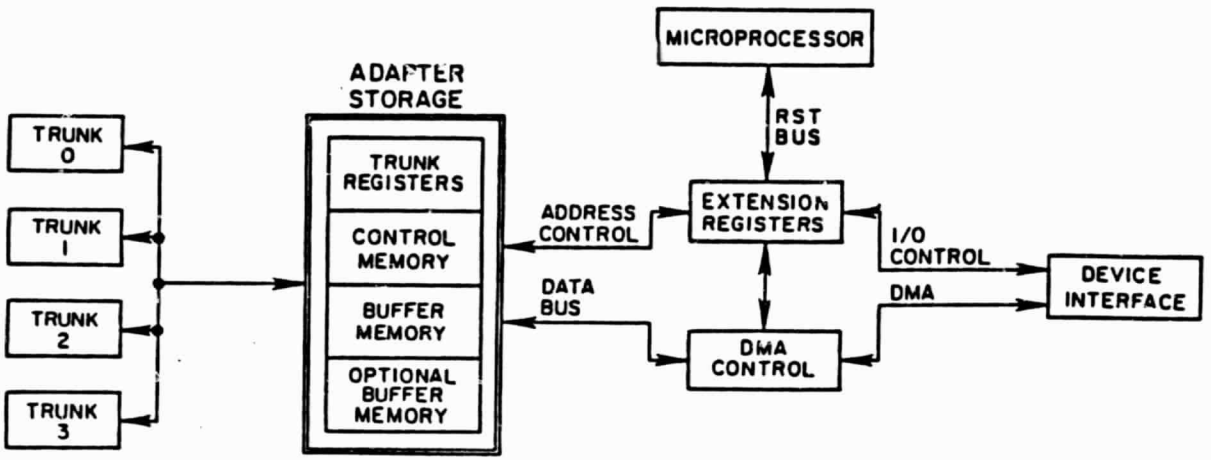


Figure 2.2 Block Diagram of Adapter [NSC80a].

trunk interface board (up to four) contains the necessary transmitting and receiving electronics and the high speed logic to control the transmission envelope-checkwords, access code, and network address. Each trunk interface has the ability to generate a busy or reserved response and contains the logic to control trunk contention on its attached data trunk. Finally, the **device interface** (up to four) or equipment interface contains the circuitry needed to interface a specific equipment or device to the adapter. This interface is unique to a given device and determines the adapter type.

2.2 HYPERchannel Protocols

HYPERchannel adapters support trunk selection, trunk-access, adapter-adapter virtual circuit, and host-adapter protocols. The trunk selection protocol is based on a cyclic scan of the connected data trunks. The trunk-access protocol is of the carrier sense multiple access (CSMA) type, with ordered priorities assigned to each adapter. The adapter-adapter virtual circuit protocol involves connection (adapter reservation and deadlock avoidance) and data flow control. The assumption of a steady state model precludes inclusion of the host-adapter protocol since a host is represented as a fixed rate data source. In order to understand the impact of the various protocols

on system performance, they will be described in detail in the remainder of this section.

2.2.1 Trunk Access Protocol

HYPERchannel uses a carrier sense multiple access scheme with prioritized staggered delays to implement a trunk access protocol. The intention of this scheme is to:

- ensure that ready adapters defer to on-going transmissions
- ensure that response (acknowledgement) frames are transmitted without interference
- provide as much trunk capacity to the highest priority adapter as it requests by prioritized access, and provide as much capacity to the next highest priority adapter as it requests, and so on
- sort out collisions without the necessity of generating random retry delays and ensuring that colliding frames are not involved in collisions when retransmitted.

HYPERchannel accomplishes this with four primary mechanisms: transmitter disable, fixed delay, N delay or priority delay, and end delay. These functions are implemented on each trunk interface board and can be different for each trunk attached.

Any time the trunk is sensed busy the adapter transmitter is disabled (unless it is transmitting) so that an ongoing transmission will not be interfered with. The **fixed delay** period is a delay period

following any transmission in which only the receiving adapter for the last transmission is allowed to transmit. This period of time allows a receiving adapter to respond immediately without being interfered with by other ready "sending adapters". A transmitting adapter which does not receive a response to a transmission during this period assumes that its last transmission was lost and will schedule a retransmission. Following the fixed delay period a scheduling period begins. This period is the **N delay** or **Priority delay** period and is a unique event for each adapter on a given trunk. An adapter's priority delay is assigned based on the priority (user defined) and placement on the trunk of the adapter in question. Following the fixed delay period an adapter ready to transmit will wait until its priority delay, at which time it is allowed to transmit without interference if the trunk is idle. If an adapter becomes ready to transmit during this scheduling period, but after its priority delay has expired it must wait until its priority delay occurs again or until the end delay period (defined next) is encountered. The **end delay** period occurs if an entire scheduling period passes without an adapter transmission. After this event a free-for-all or contention period begins in which any adapter which becomes ready to transmit may do so. In

this contention period, collisions become possible if two adapters sense the trunk idle and attempt to transmit nearly simultaneously. These various delay events are implemented in each adapter by a timer. This timer is enabled any time the trunk is sensed idle and begins to count down to its respective delay events. If at any time the trunk is sensed busy the timer is disabled and loaded with the end delay value. As the timer counts down the fixed delay, priority delay, and the end delay events are signaled (Figure 2.3).

These delay settings are set into what NSC terms as contention switches (SW1-SW9), located on each trunk interface board. These contention switches have 16 discrete positions (0 through F), with each position providing 160 nanoseconds of delay. These switches are grouped in sets of three switches per parameter. SW1, SW2 and SW3 are the least-significant bit position switches and SW7, SW8 and SW9 are the most-significant bit position switches. An adapter's fixed delay is set into switches (SW7, SW4 and SW1), its priority delay is set into switches (SW8, SW5 and SW2) and finally, its end delay is set into switches (SW9, SW6 and SW3). The actual settings for these switches are calculated from expressions given by NSC. These expressions when evaluated, yield decimal numbers which are then

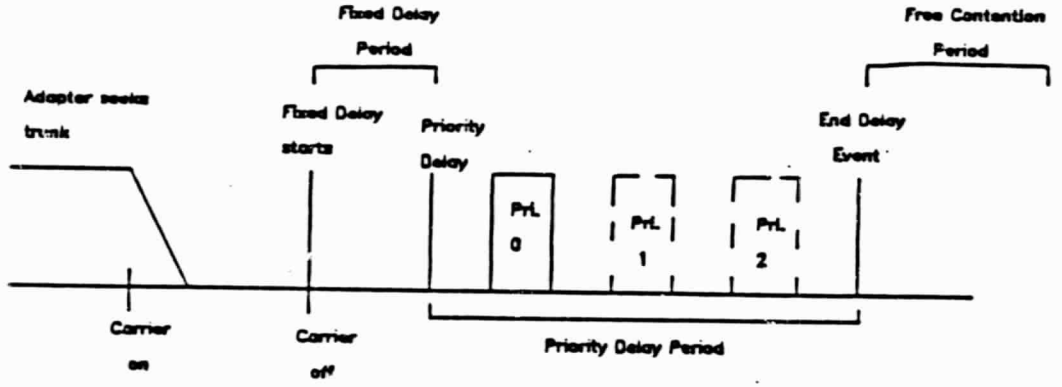


Figure 2.3 HYPERchannel Delay Time Events.

converted to hexadecimal equivalents. The hexadecimal equivalents are then set into the contention switches.

When an adapter receives a transmission frame it will immediately generate and transmit a response frame. So that all ready-to-send adapters will not interfere with this response, the fixed delay period must be such that all adapters will sense the response before this delay expires. This relation as given by NSC [NSC80a] is:

$$\text{Fixed Delay} = \frac{\text{Total trunk cable length (ft.)}}{40} + 13 \qquad (1)$$

This value as calculated will be the same for all adapters on a given trunk. The priority delay refers to the period of time uniquely assigned to each adapter on a trunk. This time period occurs following the fixed delay period and is determined by the adapter's assigned priority and position on the trunk. The highest priority adapter is assigned a priority delay of 0.48 microseconds, this allows sufficient time for the highest priority adapter to prepare to retransmit an unacknowledged frame prior to its scheduled transmission time.

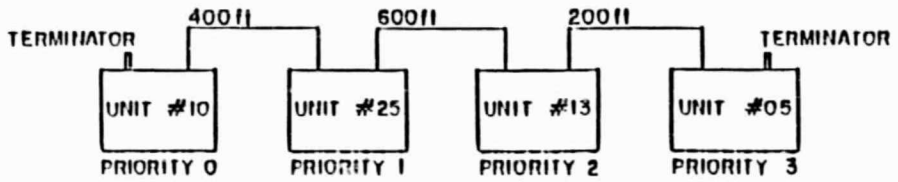
The priority delays for the remaining adapters on the trunk is given by [NSC80a]:

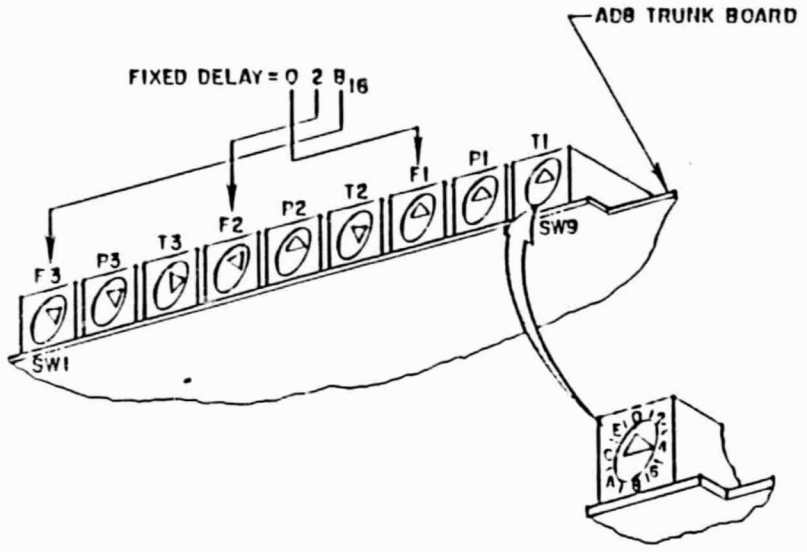
$$\text{Priority Delay} = 10 + \text{Priority delay of next highest adapter (decimal value)} + \frac{\text{Cable length (ft.) to next highest priority adapter}}{40} \quad (2)$$

NSC recommends that priorities should be assigned sequentially in order to maximize throughput on the trunk. The end delay is calculated for each adapter such that the lowest priority adapter on making a transmission is sensed by all other adapters on the trunk. This is necessary since an adapter may transmit immediately when its end delay is signaled (contention period). This delay is calculated as follows [NSC80a]:

$$\text{End Delay} = 10 + \text{Priority delay of lowest priority adapter (decimal value)} + \frac{\text{Cable length (ft.) from this adapter to farthest adapter}}{40} \quad (3)$$

An example of the use of (1), (2) and (3) is illustrated in Figure 2.4. The calculations which follow are performed for unit #25, trunk 0.





SWITCH SETTING	UNIT 10	UNIT 25	UNIT 13	UNIT 05
FIXED DELAY	28	28	28	28
PRIORITY DELAY	3	17	30	3F
END DELAY	67	50	62	67

Figure 2.4 Sample AD8 Contention Timing Calculations [NSC80a].

$$\text{Fixed delay} = \frac{1200 \text{ feet}}{40} + 13 = 43_{10} = 2B_{16}$$

$$\text{Priority delay} = 10 + 3 + \frac{400}{40} = 23_{10} = 17_{16}$$

$$\text{End delay} = 10 + 63_{10} + \frac{800}{40} = 93_{10} = 5D_{16}$$

These values are then set into the contention switches located on the AD8 trunk interface board as indicated. Finally, in the access scheme described thus far, the highest priority adapters could possibly monopolize the trunk. To prevent this a device known as a wait flip-flop is implemented in each adapter. This flip-flop is set when an adapter transmits and is cleared when the adapter end delay is signaled. When the flip-flop is set the adapter is prohibited from transmitting, the intention of this device is to provide a fair allocation of the trunk. All adapters are equipped with this device and adapters with this flip-flop enabled may coexist, on the same trunk, with adapters whose flip-flop is disabled.

2.2.2 Adapter-Adapter Link Level and Virtual Circuit Protocols

The following sections will deal with data movement and the protocols involved once an adapter has gained access to the trunk through the trunk access protocol. The data link protocol may be viewed as

having two basic information grouping levels. The basic information unit is the frame and a higher level grouping is the frame sequence.

2.2.2.1 Frame and Frame Sequence Structure

The smallest unit of information transmitted over the network is the **frame**. There are three classes of frames: **transmission frames, data frames, and response frames**. Transmission frames are generally short frames (up to 38 bytes) used by adapters for signalling and status exchanges. Table 2.1 lists these frames and their functions. There are two forms of data frames, message proper frames and associated data frames following a message proper frame (Figure 2.8). An adapter receiving a transmission frame, message proper or a data frame will generate and transmit a response frame. This response frame serves to acknowledge receipt of a frame and in some cases returns information to the transmitting adapter. Frame formats are listed in Table 2.2. Note from Figure 2.8, that the fixed delay period is used to initiate transmission of other than acknowledgement frames as will be explained.

As stated previously, a higher level information structure is achieved by grouping frames together into frame sequences. All information exchanges occurring on the network will be in the form of frame sequences. There are two types of frame sequences used on the

Table 2.1 HYPERchannel Protocol Frames [Fran 84].

Transmission Frames	
Set Reserve	- Reserve receiving adapter
Copy Register	- Request for Register information
Clear Flag 8	- When set, Flag 8 indicates that the adapter is ready to receive a frame. When clear, it indicates that the frame has been received.
Set Flag A Clear Flag A	- Flag A, when set, indicates that the receiver will receive associated data. It is cleared when the last block of associated data is received.
Clear Flag 9	- When set in the receiver, Flag 9 indicates that the receiver is ready to receive data. When set in the transmitter, it indicates that the transmitter is ready to send data. When cleared in receiver, it indicates that the data block has been received. When cleared in the transmitter, it indicates that data can now be transmitted.
Data and Message Frames	
Message Proper	- Used to transmit host messages of length less than or equal to 64 bytes.
Associated Data	- Used to transmit data blocks of length up to 2 Kbytes.
Response Frames	
Response	- A receiving adapter sends a response frame for each non-response frame received. Its purpose is to acknowledge received frames or, as in the case of the response to a copy registers frame, to send requested data.

Table 2.2 Frame Formats [FRAN82].

Field	Length (bytes)
leading syncs	12-18
frame type	1
access code	2
to address	1
from address	1
function code	2
message length	2
check word	2
trailing sync	8

Transmission Frame Format

Field	Length (bytes)
leading syncs	3
frame type	1
access code	2
to address	1
from address	1
function code	2
message length	2
check word	2
trailing	8

Response Frame Format (no data)

Field	Length (bytes)
leading syncs	12-18
frame type	1
access code	2
to address	1
from address	1
function code	2
message length	2
check word	2
message/data	8-64/ 0-2K
message/data checkword	2
trailing sync	8

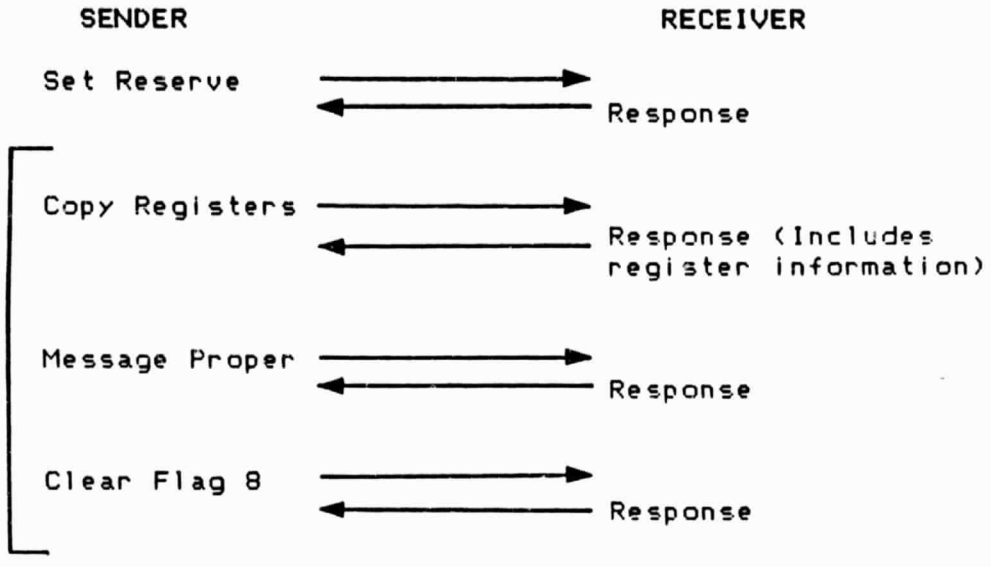
Message Proper and Associated Data Frame Format

Field	Length (bytes)
leading syncs	3
frame type	1
access code	2
to address	1
from address	1
function code	2
message length	2
check word	2
register information	2
checkword	2
trailing sync	8

Response Frame Format (with data)

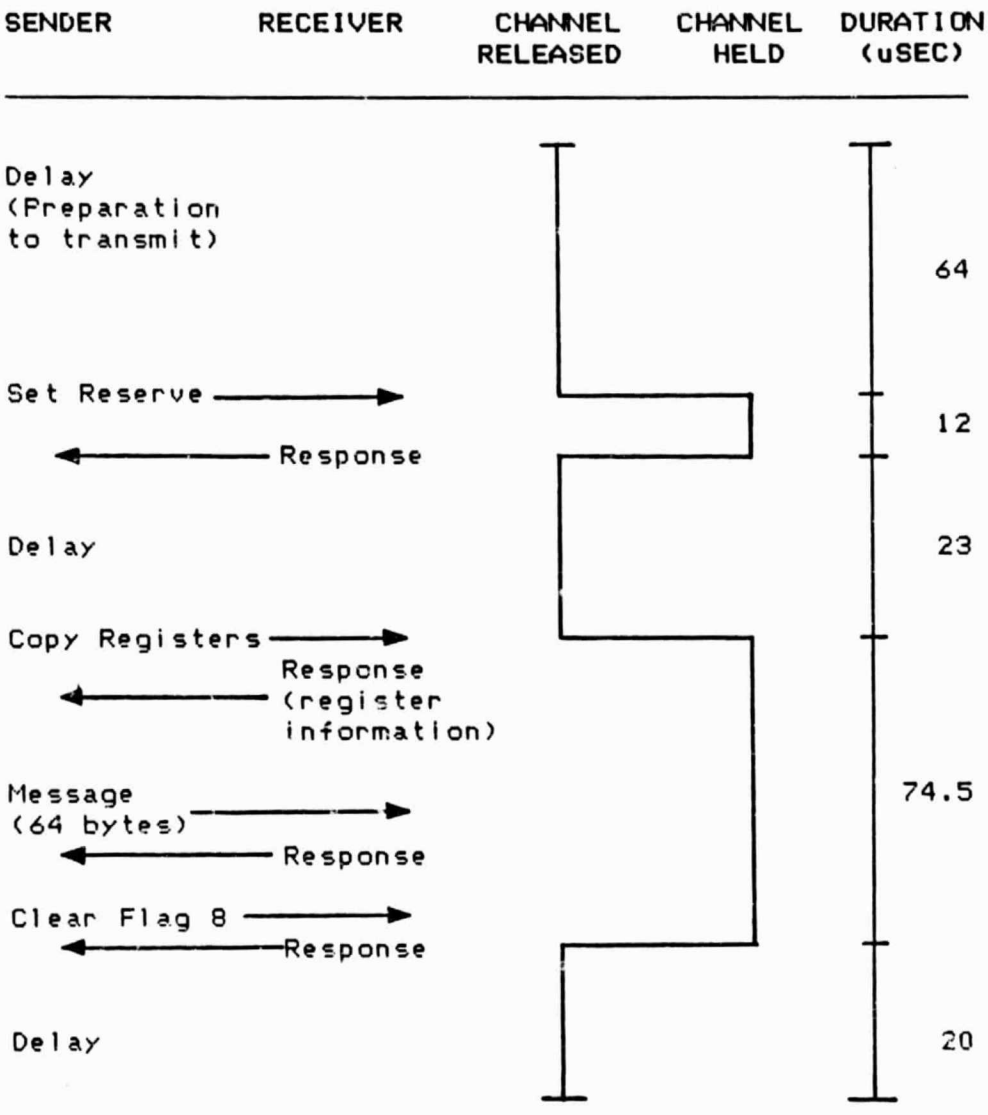
network: **message-only** and **message-with-data** sequences. A message-only sequence is normally used to transmit short messages of up to 64 bytes between devices on the network. The sequence of frames which make up a message-only sequence are shown in Figures 2.5 and 2.6. Data is transmitted on the network in 2K byte blocks (optionally 4K byte blocks) in message-with-data frame sequences, as shown in Figures 2.7 and 2.8.

In order to promote efficient utilization of the trunk an adapter does not maintain control of the trunk throughout an entire message-only or message-with-data sequence. An adapter will relinquish the trunk while performing tasks not requiring the trunk, these periods are depicted as delays in the Figures representing the two types of message sequences. After transmitting the first frame of a sequence the trunk is released. When the transmitting adapter is ready to continue transmitting it must compete with the other adapters for trunk access. After regaining access to the trunk the adapter will transmit a copy register frame, if this frame is transmitted without a collision, the receiving adapter will send a response frame to the sender. Upon receipt of a response frame the adapter captures the trunk by transmitting during the fixed delay period that follows. The trunk is regained without interference since any ready to send adapter is



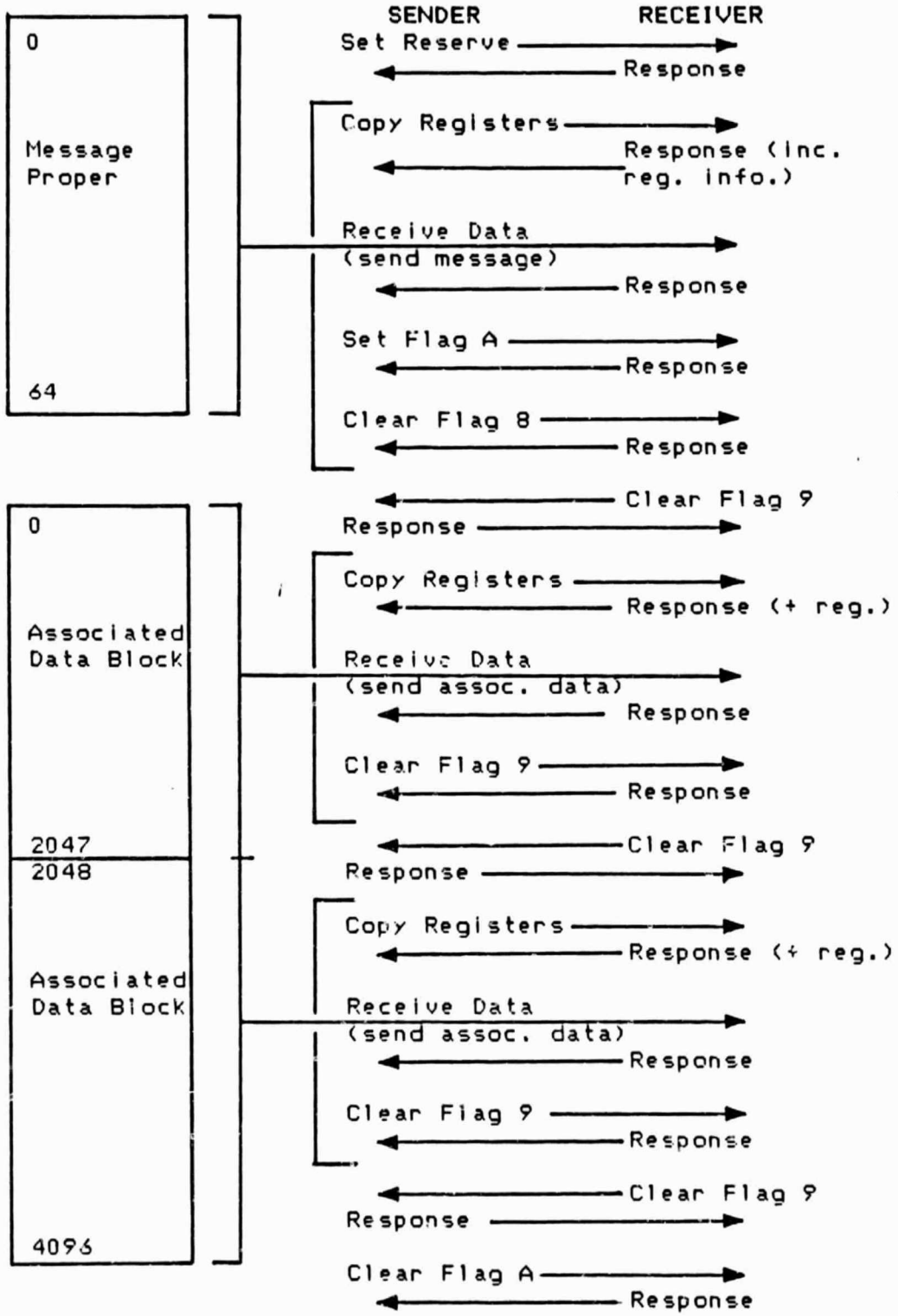
Note: Frames enclosed within bracket are transmitted without interference by beginning their transmission in the fixed delay period.

Figure 2.5 Message-Only Frame Sequence [FRAN84].



Note: Transmission times do not include propagation delay.

Figure 2.6 Timings for Message-Only Sequences [FRAN84].



Note: Frames in brackets transmitted without interference by beginning their transmission in the fixed delay period.

Figure 2.7 Message-with-Data Frame Sequence [FRAN82].

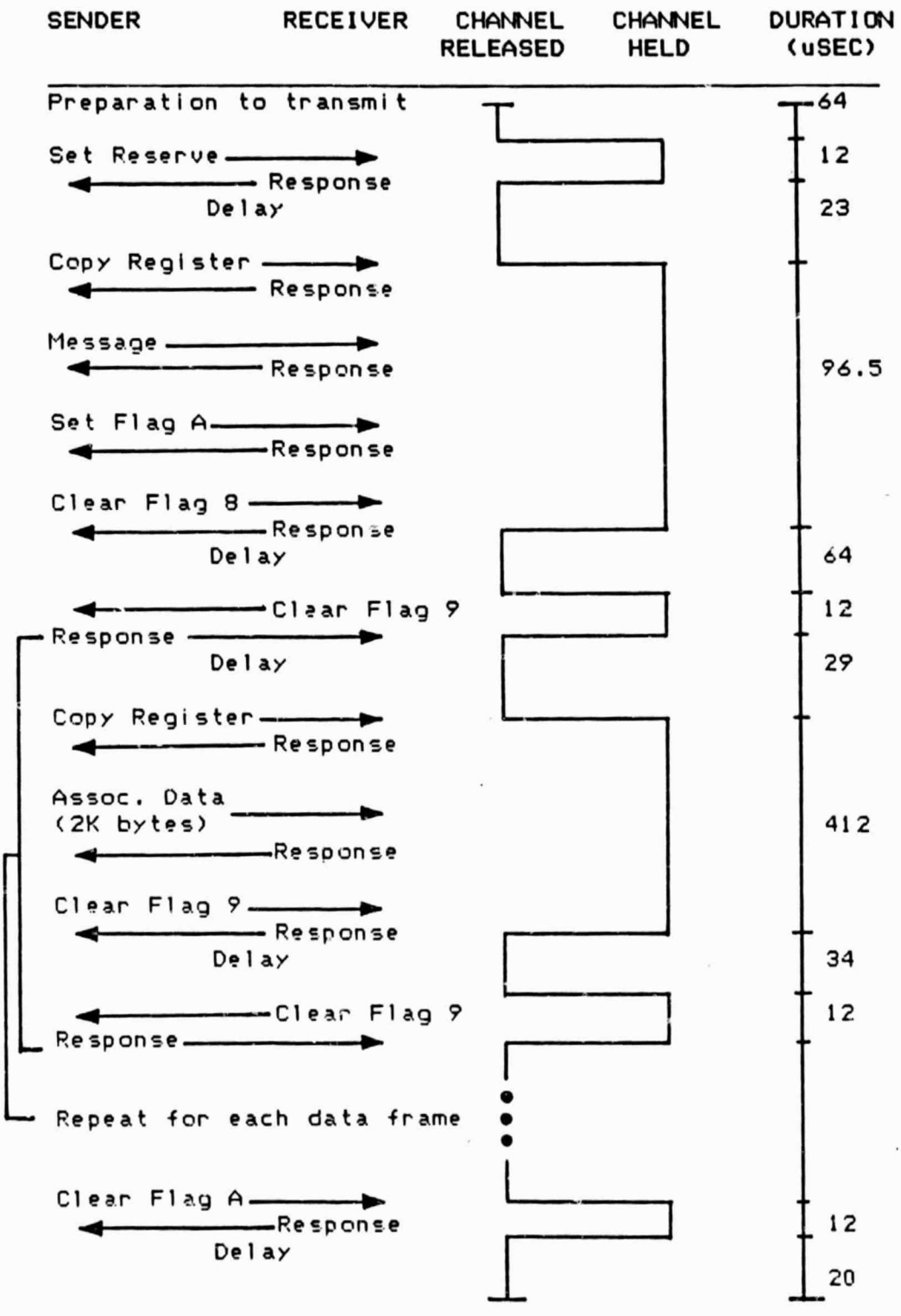


Figure 2.8 Timings for Message-with-Data Sequences [FRAN84].

prevented from transmitting during the fixed delay period. If the sequence to be transmitted is a message-only sequence, control of the trunk is not relinquished until the entire remaining sequence is transmitted, see Figure 2.6. The timing and duration of a message-only sequence is shown in Figure 2.6.

A message-with-data sequence is preceded by a message-only sequence, following the transmission of the message proper the trunk is released, see Figure 2.7. When the receiver indicates that it is ready to receive more data by sending a clear flag 9 frame to the sender, the sender will compete to regain access to the trunk. Once the sender regains access to the trunk it will send a copy register frame to the receiver. If successful, it will regain control of the trunk as described before. Control of the trunk will not be relinquished until an asscciated data frame and clear flag 9 frame are transmitted by the sender. This process is repeated for each associated block of data. Maintaining control of the trunk during data frame transmissions ensures that these frames are <u>never</u> involved in collisions, thus only relatively short frames will ever need retransmission due to collision on the trunk. The trunk is held during data frame transmission by the sending adapter, by transmitting synchronization bits during the fixed delay periods

following the transmission frames that make up the data frame sequence. The timing and duration of this sequence of frames is shown in Figure 2.8. It is important to note that in all of these sequence transmissions that during the delay periods, a transmitting adapter must always compete for trunk access in order to regain the trunk.

NSC adapters also support an alternate form of transmission scheme known as the **burst** mode. In this form of sequence transmission, an adapter will not relinquish the trunk until it completes an entire frame sequence transmission. According to literature available [FRAN82],[FRAN84] this mode is rarely used and will not be detailed in this work.

2.2.2.2 Trunk Selection

Any adapter in a network can be connected to up to four trunks. Each trunk uses the same trunk access protocol and operates independently, i.e., each trunk has a separate interface board within an adapter. The interface listens for properly addressed incoming messages and is capable of generating rejection response frames if its adapter is reserved. An adapter, having a frame to transmit and contending for a trunk will scan the trunks in a specified manner until it senses a non-busy trunk. The adapter then waits until the trunk can be captured through the trunk access

protocol. If the trunk is busy with the adapter's frame transmission its search terminates; otherwise it continues. Trunks to be tried can be specified in a host-adapter message and a receiver's trunks to try may be specified by the transmitting adapter. A trunk is said to be **disabled** if it is not connected or was not specified as a trunk to try. A trunk is said to be **enabled** if it is connected or is specified as a trunk to try. An adapter will try a trunk by first determining whether it is enabled or disabled, if the trunk is disabled or busy it will try the next trunk. If the trunk is enabled then the adapter determines whether or not it is busy. The period of time required to try an enabled trunk is 5.5 microseconds, while the time required to try a disabled trunk is 2 microseconds. The ability to specify the trunks to try in the transmitter and receiver provides a means to dedicate trunks to specific types of traffic, e.g., short transmissions on one, long transmissions on another. Simulation studies indicate no benefit is derived from dedicating trunks for particular message types. The ability to specify trunks is more useful in multitrunk networks that are not fully connected [FRAN84]. It is possible for an adapter connected to more than one trunk to receive messages simultaneously, when this happens the adapter will accept one and

reject the others. Communication between adapters on two different trunks, requires the services of an intermediary. One of the adapters will transmit to a device attached to an adapter common to both trunks. Once this device has received this message sequence it will then proceed to transmit the sequence to the intended receiver on the other trunk. Thus communication between adapters on different trunks consists of sending a message-with-data sequence twice.

2.2.2.3 Virtual Circuit Establishment

An adapter reservation scheme (Figure 2.9) is used to establish a virtual connection between adapters desiring to communicate. When an adapter receives a message from its attached host, it will first reserve itself; that is, the adapter makes itself inaccessible to received transmissions from any other adapter with the exception of the adapter with which it is trying to communicate. After reserving itself, the sending adapter transmits a set reserve frame to the receiving adapter. If the receiving adapter is idle; that is, it is not reserved, then the receiving adapter will transmit a response frame indicating that the adapter has reserved itself to receive a transmission from the transmitting adapter. Following the reservation of the receiver, the frame sequence transmission proceeds as described in section 2.2.2.1. Both the transmitting and

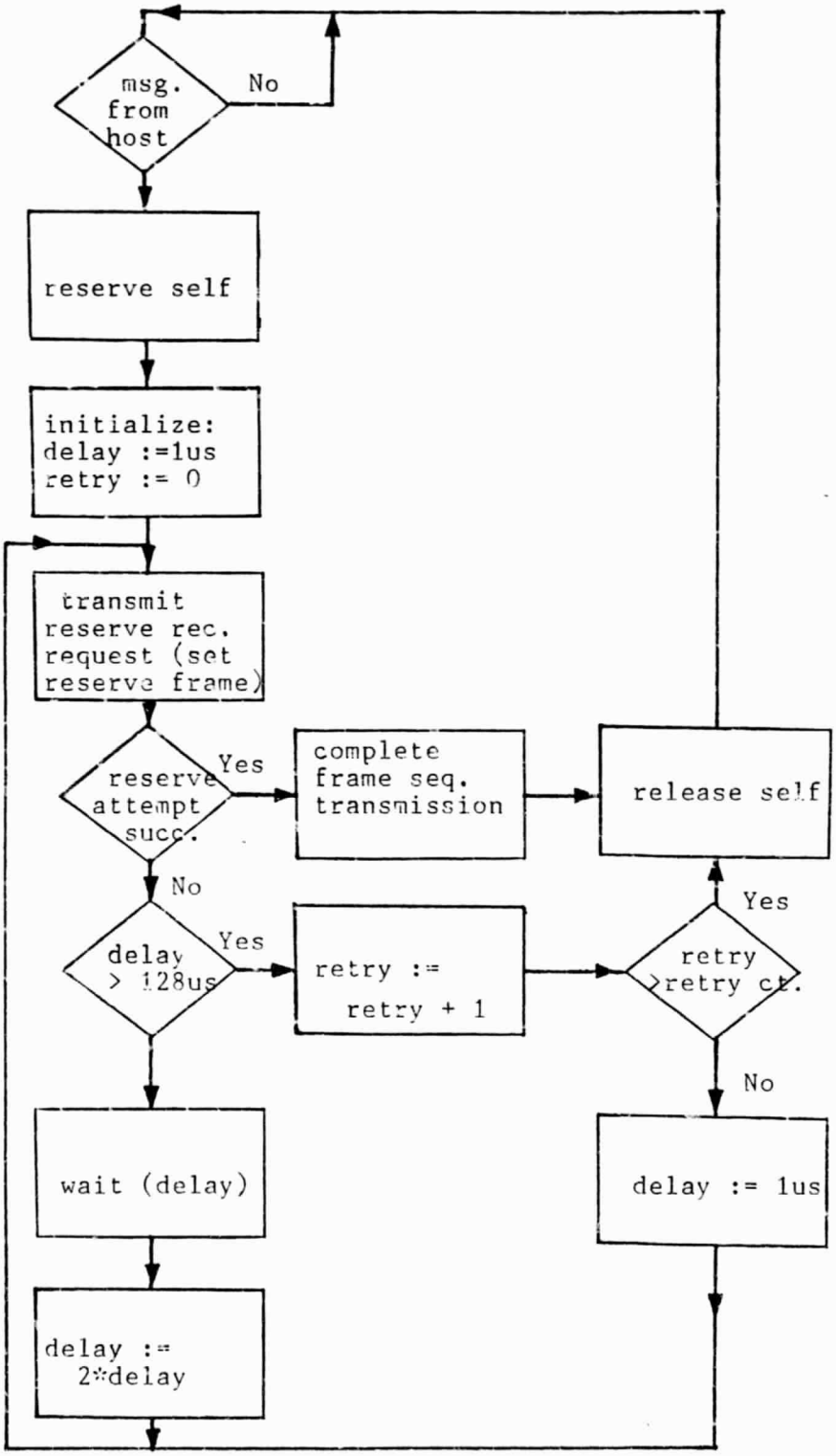


Figure 2.9 Adapter Reservation Scheme [FRAN82].

receiving adapters remain reserved during the entire message-only or message-with-data frame sequence transmission. After completing the frame sequence transmission, the transmitting adapter releases the receiving adapter with a clear flag A frame and then releases itself completing the termination of the connection; see Figures 2.6 and 2.8.

When a transmitting adapter sends a set reserve frame to a reserved adapter, the reserved adapter will respond with a reservation reject frame. The first time a sending adapter receives a reservation reject frame, the adapter will delay for a "binary exponential time" before making another reservation attempt. Following the first reservation rejection the adapter will delay 1 microsecond before making a second attempt. For each following reservation rejection, the adapter will delay an amount of time equal to twice the previous delay, up to 128 microseconds. When this value of delay (128 microseconds) is exceeded retry is advanced by one, the delay is reinitialized, and the sequence is repeated (Figure 2.9). This process will continue until either a successful reservation occurs or the adapter's retry count is exceeded. An adapter's retry count is based on its unit number, manually set by the user and is in the range of 55 to 250. If after retry count tries the adapter is unable to reserve the receiver, the

transmission is aborted and the sender returns to the idle state. The sending adapter remains reserved throughout the entire reservation attempt process. This reservation scheme with the binary exponential delays between transmissions, serves as congestion control on the network by keeping a sending adapter from monopolizing the network.

This retry scheme as described has the potential of causing adapter deadlock. This will occur if two adapters attempt to reserve each other nearly simultaneously. They will find each other reserved and go through the retry scheme until one or the other's retry count is exceeded and aborts the attempted transmission. To alleviate this problem, a back-off scheme is employed by communicating adapters. This back-off routine enables a sending adapter to determine the identity of the adapter that the rejecting adapter is attempting to reserve from the reject response. If the sending adapter determines that it and the rejecting adapter are attempting to reserve one another, then the sending adapter aborts its reservation attempt and releases itself. The slowness of this back-off mechanism can cause both adapters to abort their reservation attempts, preventing either adapter from communicating on the network. The time

required for this mechanism to respond is dependent on the adapter model and is approximately 50 microseconds.

A second form of deadlock which the back-off mechanism will not correct is called a reservation request loop. This occurs when three or more adapters attempt to reserve each other,i.e., adapter A attempts to reserve adapter B, while B attempts to reserve C, and C attempts to reserve A. In this case the adapters have no way of determining that this situation exists and each adapter will perform the reservation retry scheme as previously described. This deadlock will only be resolved when one of the adapter's retry count expires, causing the adapter to abort its reservation attempt and release itself. Serious degradation of trunk throughput may be predicted during this reservation request loop period [DONN79].

2.2.2.4 Data Flow Control

Network adapter's are supplied with two 2K byte buffers (4K byte buffers optional). With two buffers one can be filled/emptied by transfer from/to the attached host, while the other buffer is emptied/filled by a network transmission/reception. In order to ensure that buffer capacity is not exceeded, a flow control mechanism is needed. Flow control is achieved in two ways. First, data frame length is limited to adapter buffer length. Second, an adapter sending a data frame

may not send another data frame until the receiving adapter specifically requests it. This ensures that an adapter will not receive data frames at a rate faster than an adapter may process them.

Additional techniques used by adapters to prevent them from becoming permanently blocked awaiting receipt of a response frame which will never arrive are described in the following. One of the techniques involved is the requirement that response frames begin their transmission during the fixed delay period. If an adapter does not receive a response to a transmission it will reschedule the transmission at a later time. To ensure that an adapter does not retransmit indefinitely, as would be the case if the receiver address were incorrect or the receiver was malfunctioning, a limit is placed on the number of retransmissions allowed. A set reserve frame will be transmitted 16 times on each trunk and then aborted if no response is received. All other frames will be transmitted 256 times before abort. For the case when the expected response is a non-response frame (clear flag 9 granting permission to transmit data) a "deadman timer" is used. In a transmitting adapter this time frame spans the time from when the host's transmit message is received until the last buffer is transmitted and acknowledged. In the receiving adapter

this time begins with the receipt of a set reserve frame and does not end until the receipt of the last data buffer. This time span ranges from .5 to 8 seconds, adjustable in .5 second increments. If all frame transmissions are not completed before this timer expires, the transmission is aborted [FRAN82].

2.3 Intra-Adapter Communication

A message transfer between attached hosts on the same adapter is termed an intra-adapter transfer and is carried out entirely local to the adapter. This type of data transfer occurs without utilizing the data trunk network. An important consequence of this type of transfer is that the adapter will appear reserved to the rest of the network. Obviously, if the occurrence of this type of transfer becomes predominate then network performance will suffer. If resources do not conflict, intra-adapter communication may occur concurrently with network traffic and cause no degradation of network performance.

Chapter 3
HOSC System Description

As previously stated, HOSC is a distributed computer network employing many large mini-computers and NSC's HYPERchannel network. The primary task of HOSC is to provide NASA scientists and engineers with near real time acquisition and analysis of data during Space Shuttle operations. This information allows MSFC engineers and contractor personnel to act in a support capacity to other mission specialists at KSC and JSC. Some of the primary monitoring and support requirements of HOSC are pre-launch support for the Space Shuttle Main Engine (SSME), the External Tanks (ET), the Solid Rocket Boosters (SRB's), the Main Propulsion System (MPS), and the Range Safety System (RSS). Further, support is provided for mission experiments designed by MSFC personnel.

NASA has defined the mission of the Flight Operations Support Center at HOSC as follows:

> During powered flight, the HOSC will receive only data which is in the LPS (Launch Processing System) at KSC. The Shuttle support team will be in the HOSC during this phase of the mission and will be the point of contact with the JSC Mission Evaluation Room (MER) for problem discussion and resolution as required and will be on call during orbital operations. The Space Lab and experiment support team will be located in the HOSC during orbital operations when applicable.

> Following completion of the active Shuttle vehicle support activities, data is recalled as required for more detailed analysis, and initial preparation is made to provide support to post flight evaluation [NASA82].

HOSC is located in the west end of A-wing building 4663, at MSFC in Huntsville, Alabama. Figure 3.1 shows the functional components of one of many HOSC system configurations. The information contained in the following sections of Chapter 3, is from previous work published in connection with this same research effort [MAUL84].

3.1 Typical HOSC System Activities

The data processing activities at HOSC can be placed into two categories, routine daily activities and mission/launch activities. A summary of these activities by category and some details of typical data transfers are outlined in Table 3.1. While this table and the description that follows describe typical activities, these are not all the activities that HOSC is engaged in. There are some activities that are defined only for a current mission,i.e., experiments directed by HOSC mission specialists. Since these may change from mission to mission they are not included in the following description, but must be characterized and considered in the specific mission scenario required for the total system analysis.

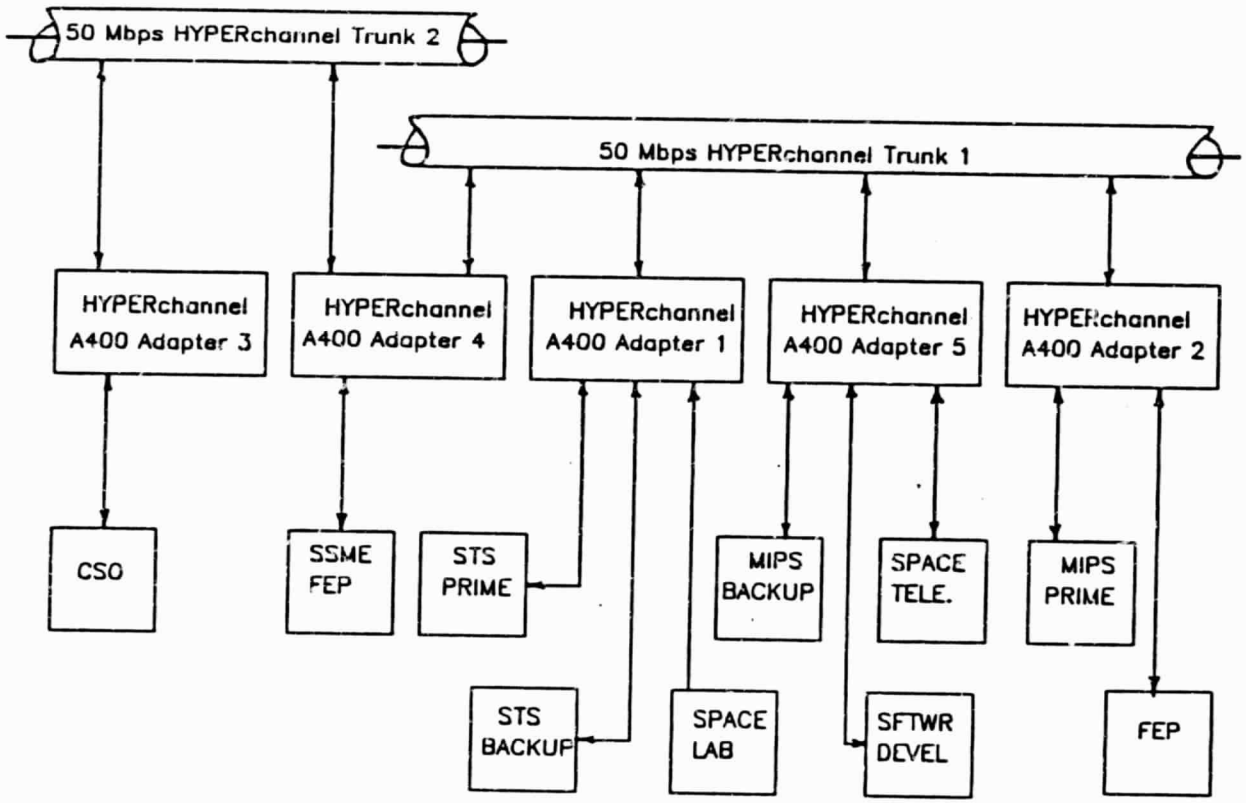


Figure 3.1 Typical HOSC Configuration.

TABLE 3.1 HOSC Data Transfers

I. ROUTINE DAILY ACTIVITIES

A. POCC Activity

Resources Involved:	MPS Primary (VAX 4) MPS Backup (VAX 1)
Quantity of Data:	150,000 512-byte blocks in two components. 6 8344-byte blocks and 100,000 512-byte blocks.

B. ECIO Data Stream (Generated by POCC)

Resources Involved:	MPS Backup (VAX 1) Spacelab 8/32
Quantity of Data:	51.2 kilobyte per second stream concurrent with POCC.

C. IGDS/SIGMA Activities (Proposed)

Resources Involved:	To be determined
Quantity of Data:	Unknown.

II. ROUTINE LAUNCH ACTIVITIES

A. Routine Daily Activities (See Item I.)

B. Main Engine Data

Resources Involved:	STS Primary (PE 3244) MPS Backup (VAX 1)
Quantity of Data:	50 kilobit per second stream (Launch minus 8 hours until MECO)

C. OD Data Stream

Resources Involved:	FEP SSME (PE 3244) CSO Computer STS Primary (PE 3244) MPS Backup (VAX 1)
Quantity of Data:	192 kilobit per second stream into FEP and then to CSO with transfer of 40 percent to STS Primary and MPS Backup. (Launch minus 9 seconds until MECO)

D. Engineering Display Changes

Resources Involved:	STS Primary (PE 3244) STS Backup (PE 8/32) Spacelab 8/32 (PE 8/32)
Quantity of Data:	Insignificant.

One activity not associated directly with the real-time responsibilities of the network and therefore a routine daily activity is POCC simulation activity. The impact of POCC activity on overall performance of the network stems from the requirement of continuous data transfers between network computers. This data is currently being transferred between the MPS primary computer (VAX 4) and the MPS backup computer (VAX 1). During each twenty-four-hour period, this activity requires the transfer of 150,000 512-byte blocks of data. This data is transferred in two components; six times each day, an 8344-byte block is transferred (50,000 bytes cumulative), with another 100,000 512-byte blocks transmitted randomly but distributed evenly throughout the day.

POCC also generates a continual 51.2 kilobit per second data stream known as the Experiment Computer Input/Output (ECIO) data stream. This data stream is ongoing and concurrent with POCC activity. ECIO data is transferred from the MPS Backup Computer (VAX 1) to the Spacelab 8/32 (PE 8/32c).

The routine components of the launch day data activities are the Main Engine Data Stream, the Operational Data (OD) Stream, and the Engineering Display Change requests. The Main Engine Data is collected and disseminated on launch day only. Data is

funneled through the network to the MPS Backup Computer (VAX 1) from the Shuttle Transport System (STS) Computer (PE 3244). From eight hours before launch until about twelve minutes after launch at Main Engine Cut Off (MECO), STS Primary accepts a continuous 50 kilobit per second data stream directly from the KSC Firing Room and transfers about 24 percent (12 kilobits per second) of the total stream to MPS Backup.

The OD Stream is a 192 kilobit per second data stream arriving at the Front End Processor Space Shuttle Main Engine (FEP SSME) Computer (PE 3244) on launch day concurrently with some of the Main Engine Data (launch minus 9 seconds until MECO). This data arrives from Goddard Space Flight Center in Maryland, and is transferred over the network to the CSO facility.

The Engineering Display Change activity is an almost insignificant addition to the total network traffic. This activity involves a transfer from STS Primary to STS Backup and Spacelab 8/32. This transfer consists of the name of each engineering console format in the support facility that is changed during the prelaunch and launch period (launch minus 9 seconds to MECO) [MAUL84].

3.2 HOSC System Components

The Huntsville Operations Support Center as a distributed computer facility uses various computer resources interconnected together. The basic components involved at HOSC are the computers, communications processors that receive data from wideband communications links (satellite and terrestrial) and the hardware required to interconnect these various resources. The interconnecting hardware, Network Systems Corporation's HYPERchannel network, has been previously described, and interconnects a variety of computer resources of different manufacture, i.e., Digital Equipment Corporation (DEC), Perkin-Elmer (PE), and UNIVAC. The bulk of the processing power for the network is performed by the DEC VAX and PE 3240 computers. A cursory look will be given to these two computers. These computers are examined for the sake of information only, since in the subsequent analysis all network users will be represented as fixed rate data generators.

The Digital Equipment Corporation's VAX series of computers supports a 32-bit word architecture that establishes a virtual address space of 4.3 billion bytes of user-addressable memory. Throughput of data in the system is optimized by using a 32-bit high speed data structure that ties together the central

processor, main memory, UNIBUS subsystem, MASSBUS subsystem and the DR780 high speed direct memory access subsystem. This high speed data structure is known as the Synchronous Backplane Interface (SBI) and a device linked to it is known as a NEXUS. Each NEXUS receives every SBI transfer; electronic logic in the NEXUS determines whether it is the designated receiver for the ongoing transfer. Data transfers can occur from CPU to memory, from I/O controller to memory subsystem, or from CPU to I/O controller, with maximum aggregate transfer rates of 13.3 megabytes per second. These transfer rates are limited by the following factors:

- 200 nanoseconds/cycle = 5 million cycles per second
- each cycle can carry an entire byte of data or address representing a memory request
- one cycle is used to request eight bytes of data to be read or written, and two cycles are used to carry data at four bytes per cycle
- 5 million cycles/second * 4 bytes/cycle = 20 million bytes/second
- 20 * 2/3 (1 of every 3 cycles is an address) = 13.3 million bytes per second [DEC80]

As is shown in Figure 3.2 , a VAX computer may interface with more than one memory subsystem. In the case of a two-controller, interleaved memory

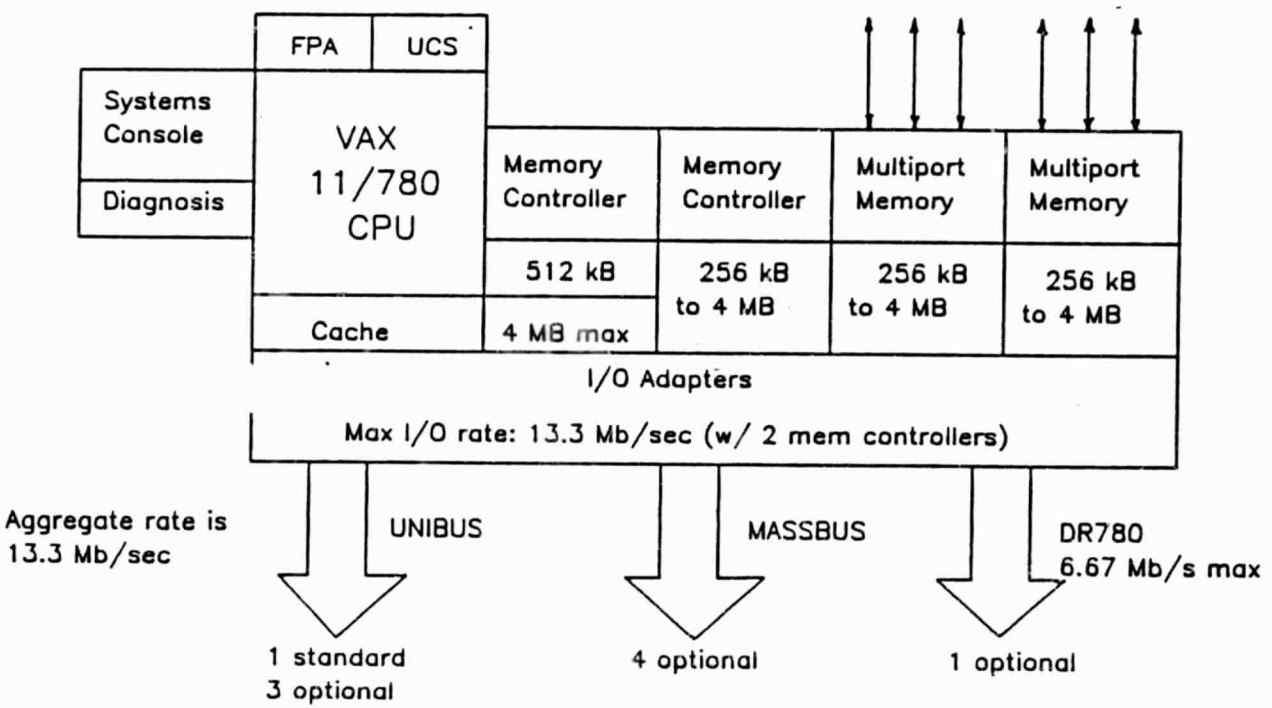


Figure 3.2 Block Diagram of VAX 11/780 Computer.

configuration, the computer would also have two NEXUS memory controllers on the SBI.

The UNIBUS Subsystem (UBUS) is a high speed, asynchronous data system that allows communication between peripheral hardware and the VAX computer. The VAX 11/780 is capable of supporting four UBUSes: one is standard, but three more are optional (Figure 3.3). The UBUS is connected to the SBI through a UBUS Adapter (UBA) which performs several important services that are transparent to the user. The UBA provides:

- access to UBUS address space from the SBI
- mapping of the UBUS addresses to SBI addresses for the direct memory access (DMA) transfers to the system memory
- data transfer paths for UNIBUS device access to random SBI memory addresses and high speed transfer for devices that transfer to consecutive, increasing addresses
- UNIBUS interrupt fielding
- UNIBUS priority arbitration

The address-mapping function is especially important because the UBUS has only eighteen data lines providing an apparent memory-addressing capability of only 200 Kilo-bytes. Through its address mapping abilities, the UBA provides the capability of mapping the UNIBUS adoresses into the SBI addresses so that the

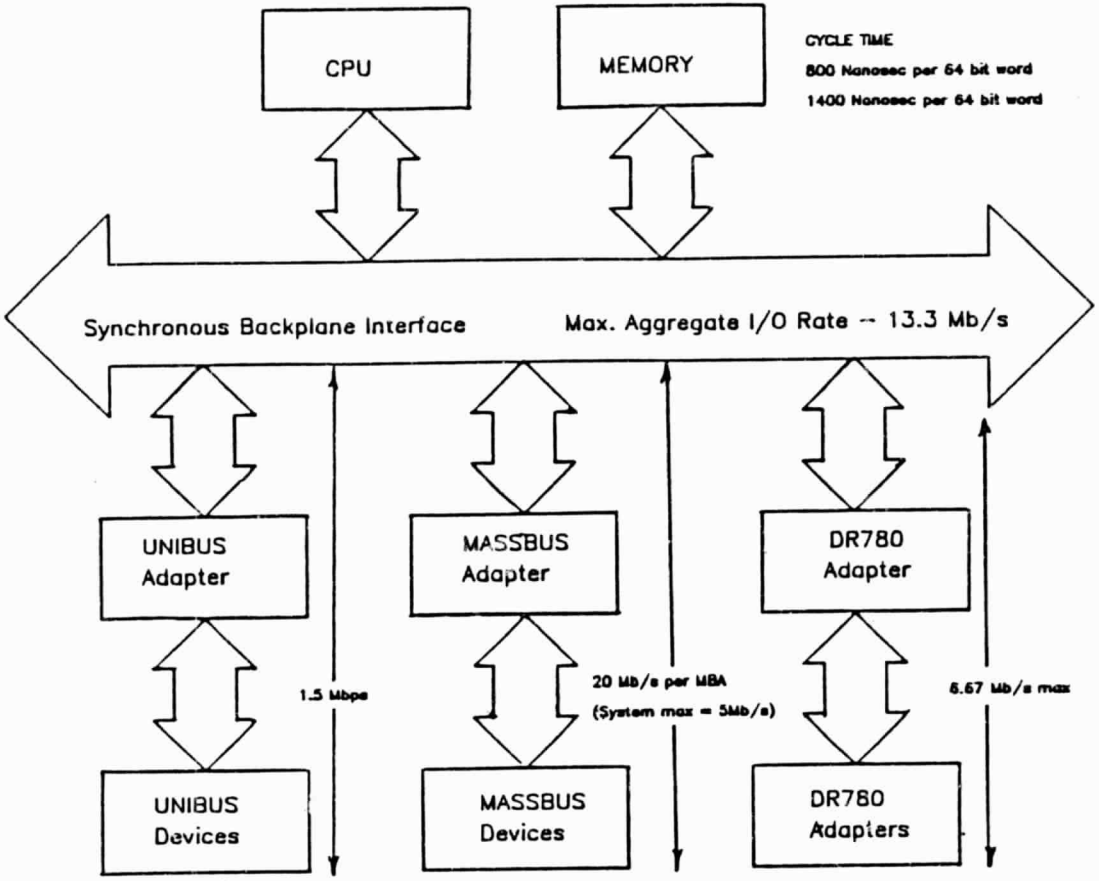


Figure 3.3 Bus Configuration of VAX 11/780.

full system memory of 16 array boards of 256 kilobytes each (4 gigabytes total) can be accessed.

The UBA accepts two forms of input from the UBUS: hardware-generated interrupts and direct memory access requests. Each device connected to the UBUS may use one of five priority levels for requesting bus service. A non-processor request (NPR), which is the highest priority request, may be used when the device requests an operation that does not require processor intervention, such as a direct memory access transfer to memory or to some other device. A bus request (BR) is thus generated when the device wishes to interrupt the UBA for service. Such service might be a CPU-directed data transfer or a flag indicating the existence of an error condition at the peripheral. Since there are only five priority levels and more than one device may be connected to a specific request level, if more than one device makes the same request, the device that is electrically closest to the UBA receives the highest priority.

The NPR request for direct memory access is a very important feature of the UBUS subsystem. These DMA transfers can be divided into two groups: random access of noncontiguous addresses and sequential access of sequentially increasing addresses. For random access, each UBUS transfer is made through the Direct Data Path

(DDP, one per UBUS) and is mapped into an SBI transfer. This procedure allows only one word of data to be transferred during a single SBI cycle. For devices capable of requesting sequential access services, use is made of the Buffered Data Path (BDP). Each UNIBUS provides fifteen such BDPs which store the data so that four UBUS transfers are performed for each SBI transfer.

The DDP must be used by devices not transferring to consecutively increasing addresses or by devices that mix read and write functions. The maximum throughput via the DDP is about 425 kilowords per second for write operations and about 316 kilowords per second for each read operation. These rates will decrease as other SBI activity increases [DEC80].

Maximum published throughput via the BDP is about 695 kilowords per seconds for both the read and write operations, but realistic throughput rates of only 1.5 megabits per second are actually expected with this figure degrading as other SBI activity increases. BDP transfers are restricted to block transfers with blocks of length greater than one byte. All transfers within the block must be to consecutive and increasing addresses and all transfers must be of the same function, either read or write.

The MASSBUS subsystem and the DR780 high performance, 32-bit parallel interface will not be described in this summary, since an understanding of their functional characteristics is not needed to determine their relative impacts on the HYPERchannel network. The influence of both may be felt indirectly, however, since activity on the MASSBUS or DR780 will translate to SBI activity which will affect DDP and BDP transfer rates as previously described.

The VAX CPU will also not be described in detail but several comments may be made about the CPU and its effects on system throughput. The CPU represents the most intensive traffic load on the memory subsystem and hence on the SBI. Obviously, it the processor is engaged in intensive computing, it will request data much more often than it will write data, and this memory access represents substantial SBI activity. Fortunately, the large cache memory of eight kilobytes available to the CPU is able to reduce the SBI traffic load considerably. In addition to the effects of the CPU on SBI activity, the SBI traffic from other sources may also affect the CPU efficiency. Published figures, [DEC80], indicate that in a system with two memory controllers the processor will be slowed by about four percent per averaged megabyte per second of I/O traffic, while the impact of a single memory controller

is to slow the processor by a factor varying from two to four. Table 3.2 summarizes the I/O characteristics of the DEC VAX 11/780 processing system.

The Perkin Elmer 3240 series of computers is also a high throughput machine with a 32-bit architecture. The HOSC currently uses two PE 3244 machines with primary responsibilities as front end processors receiving real-time data streams from the KSC firing room.

The 3244 memory subsystem is organized into banks, each capable of handling four megabytes of addressable memory. Total system memory ranges from 256 kilobytes in one bank to a full system complement of four 4-megabyte banks, for a maximum of 16 megabytes of addressable memory. All memory is connected to a common memory bus which consists of two unidirectional, asynchronous, 32-bit busses. One bus is dedicated to memory write functions and the other is dedicated to memory read functions (Figure 3.4).

Input/Output is accomplished using five external communication busses: one multiplexer bus for medium speed devices and a maximum of four high-speed Direct Memory Access (DMA) busses that each support eight high speed bidirectional ports. Each DMA port is controlled by a selector channel tied to the multiplexer bus that controls and terminates transfers through the CPU. Once

TABLE 3.2 Summary of DEC VAX 11/780 I/O Characteristics

Processor:	32-bit word architecture
Main Memory:	
Virtual address space:	4.3 billion bytes.
Cycle Times:	800 nanoseconds per 64 bit read. 1400 nanoseconds per 64 bit write.
I/O UNIBUS Adapter	
Maximum aggregate rate:	1.5 Mbyte/sec through BPD
Buffered Data Paths (BPD):	15 total, 8 byte buffer in each. 695 kilowords per second read and write. Used for fast data transfers.
Direct Data Path (DDP):	425 kilowords/sec for write and 316 Kilowords per second for read. Used for transfers to nonconsecutive memory locations.

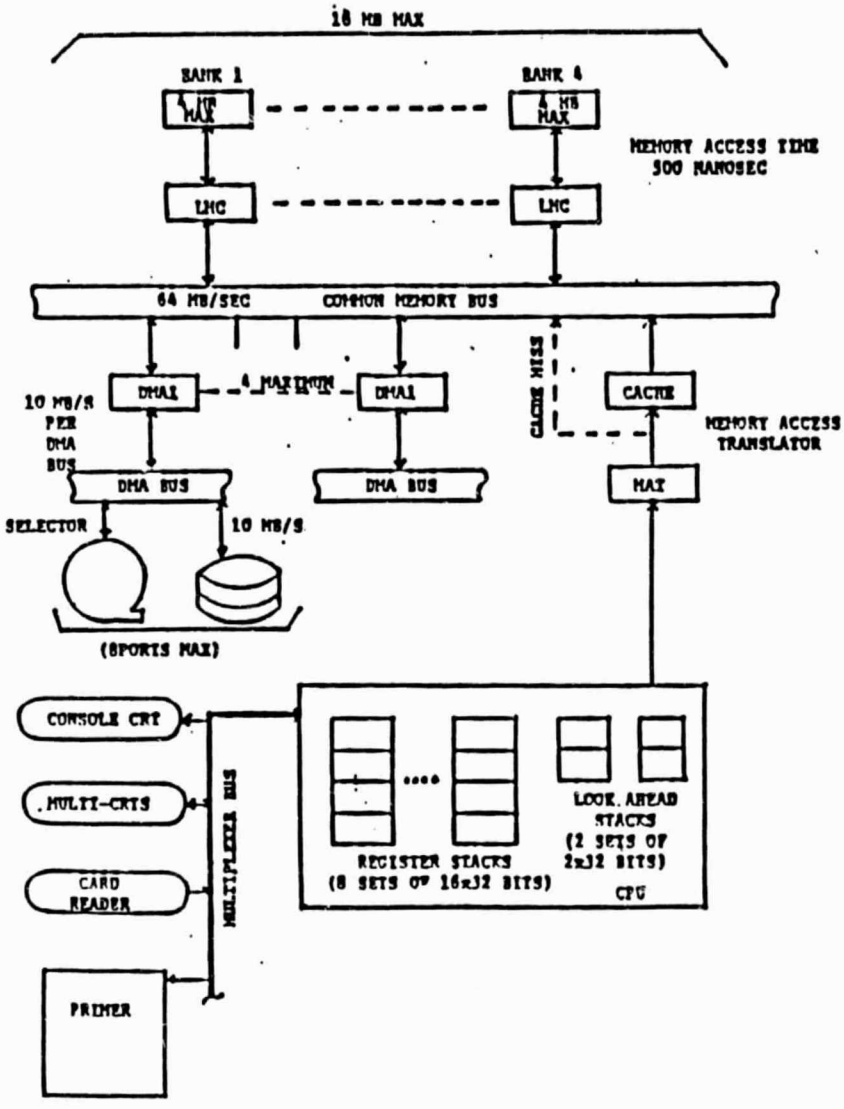


Figure 3.4 Block Diagram of PE 3244.

the channel is activated, the processor is released and is free to continue processing on an unrelated task. Published I/O transfer rates for the PE 3244 DMA bus indicate that transfer rates of up to 10 megabytes per second in the burst mode are possible for each DMA bus [PEC81]. The I/O characteristics of the PE 3240 series is summarized in Table 3.3 [MAUL84].

TABLE 3.3 Summary of Perkin Elmer 3240 I/O Characteristics

Processor:	32 bit/word.
Main Memory:	
Virtual Address Space:	16 Megabytes.
Cycle Time:	500 nanoseconds.
DMA Bus Data Transfer Rate	
Burst Mode:	10 Megabytes per second with a maximum of 4 DMA busses per system.

Chapter 4
Software Design Description

This chapter describes the development of the software used to model the HYPERchannel network. The intent of the simulation is to accurately model the characteristics of a HYPERchannel network. While, every possible function of the network is not characterized, the protocols which have the most impact on the network were included to the extent possible. The goal of the simulation is to provide a software tool to analyze the characteristics of present and future configurations of the network at HOSC.

4.1 Software Functions

The simulation model as developed should perform some primary functions, among these are:

1) provide a relatively easy method of simulating the existing system and provide a means for modifying the simulation environment so that different configurations may be modeled.

2) provide an output which may be meaningfully converted to performance measures described in this Chapter and Chapter 5.

3) provide a user friendly interface at all levels of user interaction.

The software should be designed to be as portable as possible so that transferral from the development computer to other computer environments should be accomplished as easily as possible. Also, any future

enhancements of the algorithm as designed, should have the capability of being accomplished as easily as possible.

Finally, the development of this simulation as related to the ISO-OSI model for distributed computer processing (Appendix II), may be thought of, as having been developed between levels 5 and 6, the Session and Presentation levels.

4.2 Design Constraints

The primary design constraint placed on this simulation effort was that the simulation be as transportable as possible between computers of unspecified size and capabilities. Therefore, the language PASCAL was chosen due to its widespread availability and richness of data structures.

4.3 Design Specifications

This section deals with the imposed constraints and assumptions which the model uses to accomplish its simulation activity. These assumptions were made to keep the model both realistic and manageable.

4.3.1 Simulation Assumptions

In order to maintain viability and model validity certain assumptions were made. These assumptions are as follows:

1) Every time an adapter requests the trunk, the frame sequence sent is a message-with-data sequence. Therefore, message-only sequences are excluded.

2) Maximum number of trunks is two.

3) Model will reduce to a single trunk model.

4) Any single adapter may be attached to both trunks (dual trunk model) (1,2,..,N)

5) Trunk lengths are less than, or equal to 1000 feet, Belden 9248 cable data is used (propagation time).

6) Worst case fixed delay, priority delay, and end delay will be used, as given by [FRAN84].

7) In order for the model to resemble reality as closely as possible, measured transmission frame times will be used ([WATS82],[FRAN82]).

8) Intra-adapter communication is not allowed.

9) The wait flip-flop option is not implemented.

4.3.2 Parameter Definition

As previously stated worst case values of fixed delay, priority delay, and end delay are used in the simulation. The expression used to evaluate the fixed delay on a given trunk is given by:

$$\text{Fixed delay} = 4 \text{ nsec} * (\text{trunk length}) + 2.08 \text{ usec} \quad (1)$$

where trunk length is given in feet. This value approximates, but is greater than, twice the end-to-end

propagation time for the trunk, plus the time required by the adapter to formulate and to begin transmitting a response. This value is the same for each adapter on a given trunk.

The highest priority adapter is assigned a priority delay of 0.48 usec. The priority delay for the remaining adapters on a given trunk is given by:

$$\text{Priority delay (K)} = \text{Priority delay (K-1)} + 4 \text{ nsec} * d + 1.6 \text{ usec}, \quad K = 2,3,....L \qquad (2)$$

where K is the index of the adapter with priority K and d is the distance in feet between the adapter with priority K and the adapter with priority (K-1), L refers to the lowest priority adapter. The constant 1.6 usecs is added to alleviate problems with line reflections.

The end delay of the adapter with priority K is given by:

$$\text{End delay (K)} = \text{Priority delay (L)} + 4 \text{ nsec} * d + 1.6 \text{ usec}, \quad K = 1,2,....L \qquad (3)$$

where L is the lowest priority adapter, and d is the distance, in feet, from the adapter with priority K to the adapter farthest from it [FRAN84].

These are the major parameters which need to be calculated for a given configuration of the HYPERchannel network. They are calculated for a given configuration when the simulation program is executed.

4.4 Algorithm Description

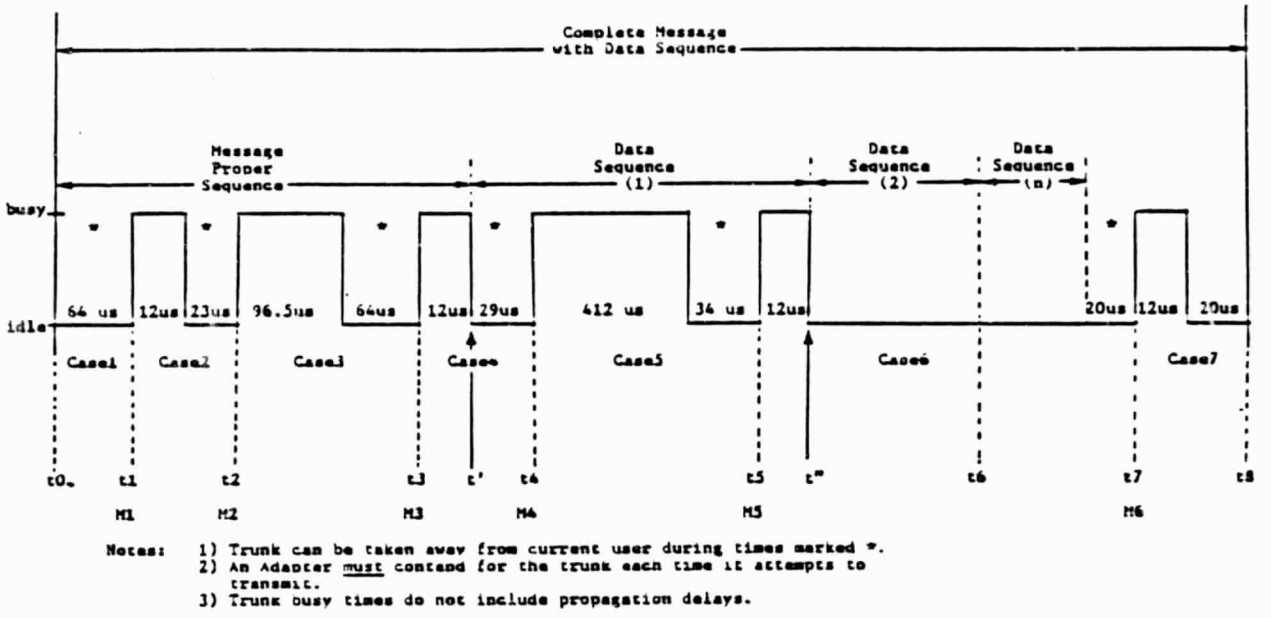
In this section an overall view of the activity of the simulation is detailed. Detailed descriptions of the individual modules used by the program are found in Appendix III.

4.4.1 Overall Simulation Activity

The simulation as developed is a discrete event simulation. The event that drives the model is an adapter's next scheduled transmit time. An adapter trying to transmit will attempt to obtain a trunk in three given time frames. An adapter may try to transmit while the trunk is busy, during the priority delay or scheduling period, or finally, in the contention period. If an adapter attempts to transmit while the trunk is busy its next transmit time is adjusted to its scheduling period time (priority delay). If an adapter attempts to transmit during the scheduling period one of two possibilities exist: 1) it attempts to transmit after fixed delay but before its priority delay event and 2) it attempts to transmit after its priority delay event but before the end delay event. In the first case, the adapter is allowed to transmit at its

priority delay event and in the second case, the adapter is forced to wait until its end delay event is signalled. This is in accordance with NSC's trunk access protocol. An adapter attempting to transmit during the contention period is allowed to do so provided that the trunk is sensed idle. Once the adapter has gained access to a trunk through the trunk access protocol as modeled, it will proceed to send its message sequence.

In Figure 4.1, a complete message-with-data sequence is shown. Appropriate timing and program flags (M1 through M6) are indicated. the algorithm shown in Figure 4.2, in flow diagram form, illustrates the overall activities of the simulation program. Using these two figures a basic description of the program flow may be performed. The first three blocks in Figure 4.2 perform initialization of program variables, definition of the network configuration, and prints a description of the network to the auxiliary file Auxout maintained by the program. Once trunk access has been granted (previously described) an adapter will attempt to send transmission frames and data frames. Program flags are used to indicate where in the message-with-data sequence the present transmitting adapter is located. The various update blocks (Case1 through Case7) update timing, bit counts, and flags.



Notes: 1) Trunk can be taken away from current user during times marked *.
2) An Adapter must contend for the trunk each time it attempts to transmit.
3) Trunk busy times do not include propagation delays.

Figure 4.1 Message-with-Data Sequence for Simulation.

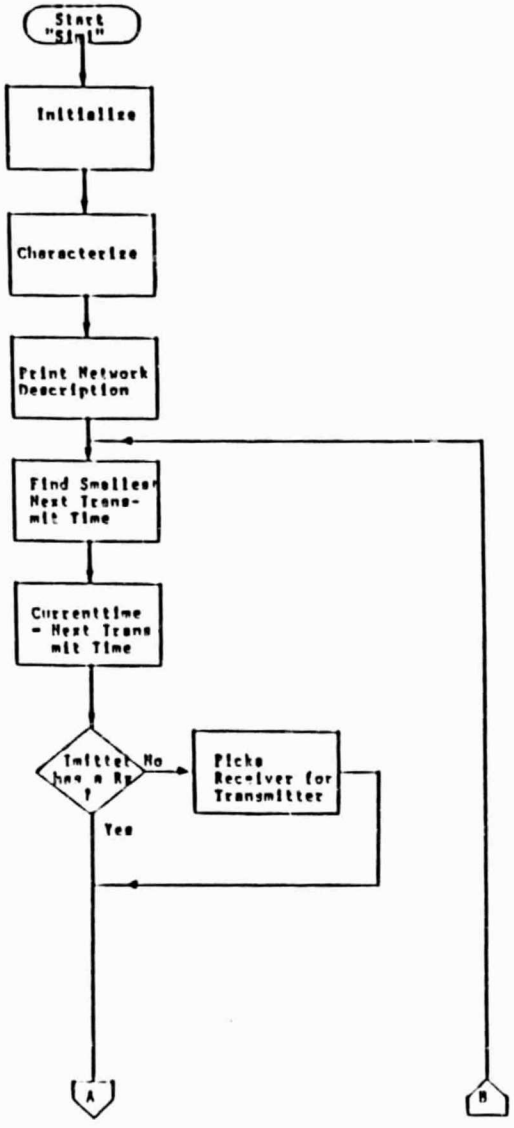


Figure 4.2 Flow Diagram of Simulation Activity.

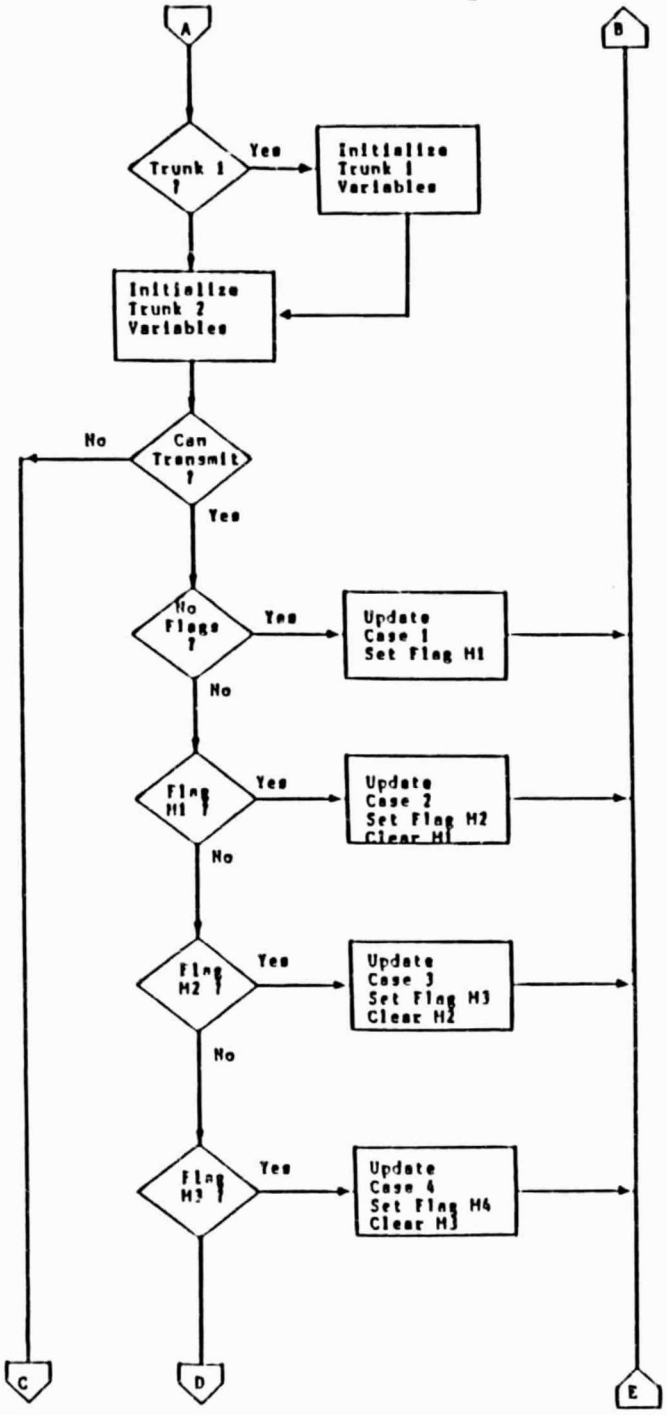


Figure 4.2 Continued.

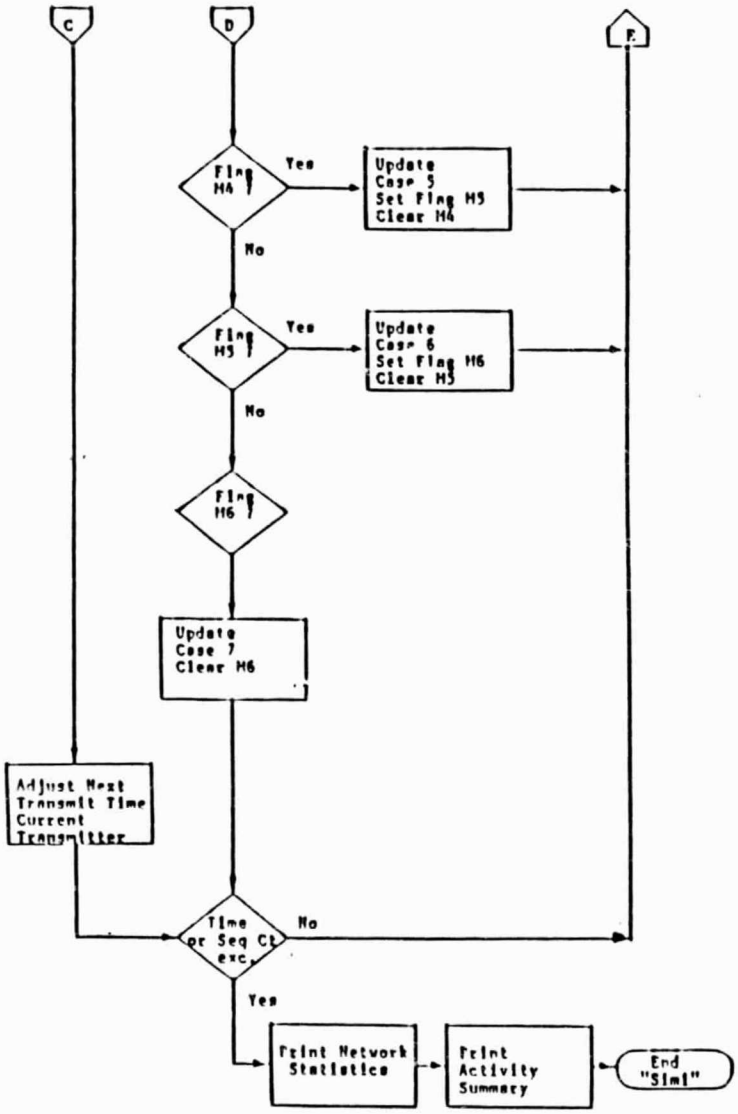


Figure 4.2 Continued.

Case2 incorporates the adapter reservation scheme (Figure 2.9). If a receiving adapter can be reserved for the transmitting adapter the message sequence transmission will continue, otherwise, the adapter will return to case2 each pass through the loop, until it reserves its receiver or aborts its transmission. The message proper frame is updated in case4. A message length of 64 bytes is assumed for convenience. Case5 updates the transmission segment in which a data block is sent. In case6, determination of whether or not another data sequence needs to be sent (based on the attached device's buffer size) is made. If another data block is required, then the period of time t'- t" will be repeated, this will continue as required. In case7, updates to message delay (the time difference between initial transmission time and the final transmission time), trunk sequence transmission tally, and adapter next transmission time are performed. The transmitting and receiving adapter's are released for other transmissions in case7 also. The simulation will continue until the user defined maximum time or the maximum number of message sequences is exceeded. The simulation will then print output statistics to an interactive terminal and an output file.

4.4.2 Model Set-Up

Figure 4.3 illustrates the basic set-up of the network configuration which the simulation program operates on. All configurations to be modeled should be set up according to this diagram. In the case of a single trunk configuration, adapters 1 through N are used.

The simulation (Sim1) uses this model configuration to properly identify and characterize the network adapter's, their attached devices, and data sources.

4.5 System Files

There are three files associated with the simulation: program, Dfile, and Auxout files. The program file (Sim1) may be maintained as either source or object code since normal operation of the program requires no changes to the program file. The files Dfile and Auxout are maintained as standard ASCII text files. Dfile contains a description of the network, and is used by the program file for parameter values. Auxout contains a text description of the network and is used to store output statistics from a particular program execution.

At the beginning of a program execution, an interactive query allows the user to use the network configuration defined in the Dfile or define a new

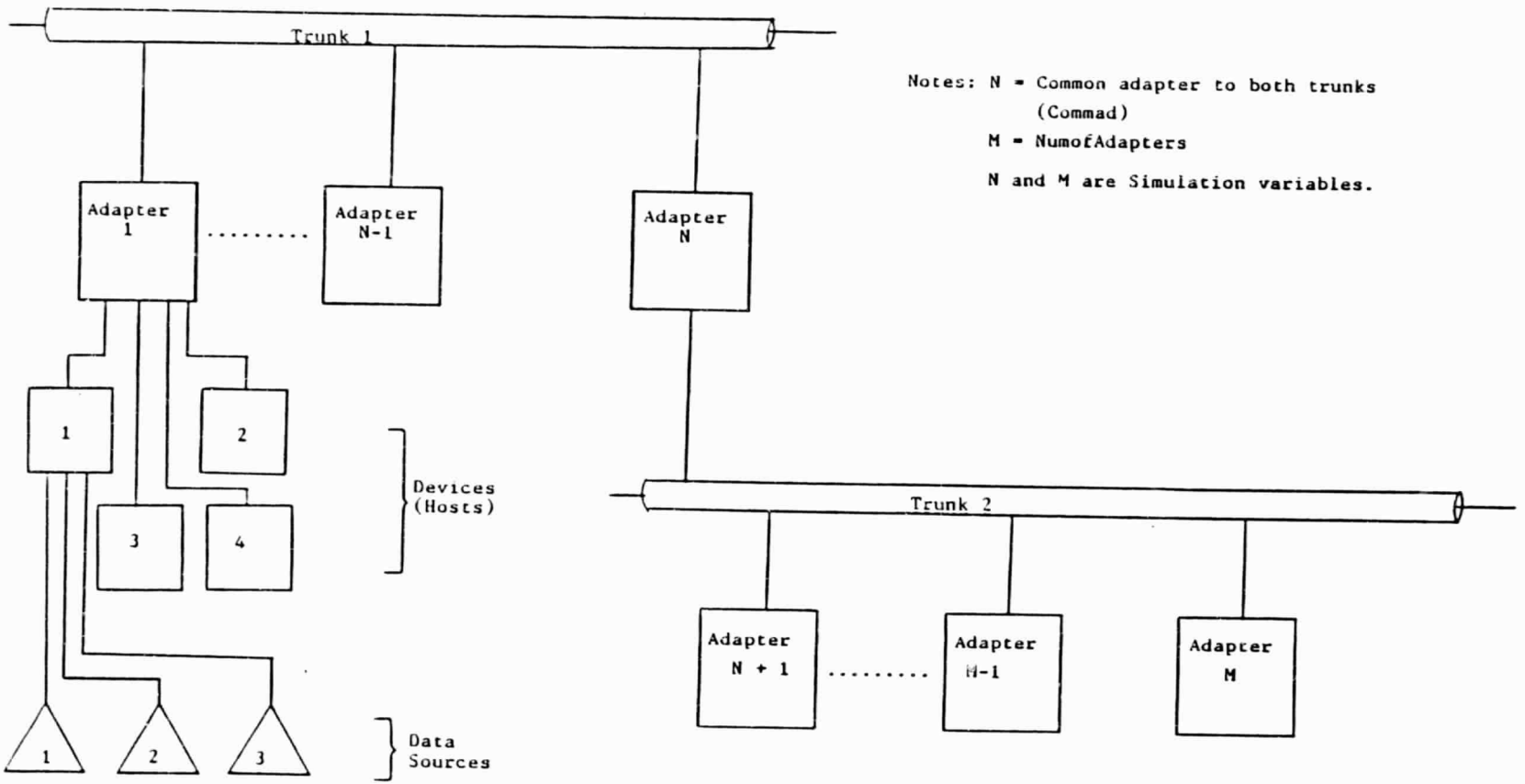


Figure 4.3 Model Set-Up for Simulation.

configuration, which is then placed in this file. The auxiliary file Auxout, is a print formatted file and may be directed to a printer for a hard copy of a program execution's statistics. Additionally, Auxout is totally maintained by the program file and thus is transparent to the user until a print out of a simulation run is desired.

4.6 Reported Statistics

The network statistics, as compiled by the model, are such that they allow the computation of measures which indicate network performance. These measures are defined as follows:

- D: The delay that occurs from the time an adapter becomes ready to transmit a message sequence until it actually completes its transmission.
- S: The throughput of the local network; reported as the total rate of data transmitted over the network.
- U: The utilization of the network; this is given as normalized throughput.
- G: Offered load; the total rate of data presented to the network.

where possible the statistics necessary to calculate these measures will be reported for the entire network, the individual trunks, and the individual adapters.

4.7 Validation Criteria

While every attempt has been made to make this simulation as close to the real system as possible, software implementation of an electronic system has its limitations. This simulation effort has been aimed primarily at modeling the various protocols implemented by a HYPERchannel network. It has been shown that these protocols are the factors having the most impact on system performance [FRAN84], [WATS80], [DONN79], and so on. Therefore, the simulation was designed to model these parameters as closely as possible. The validity of this model rests on a clear understanding of the operation of HYPERchannel networks and the relationship of this operation on system performance.

Finally, it was not feasible or practical to establish a validation test bed at HOSC. This due largely to the requirements placed on the facility and its stated mission. Thus, the model validation was done empirically, based on knowledge of the HYPERchannel system and the components attached at HOSC.

Chapter 5
Model Analysis and Conclusions

In this chapter some of the theoretical considerations involving HYPERchannel network performance measures will be developed. The results of exercising the model in various configurations (based on the HOSC system and components) will be presented. Finally, the conclusion section, will attempt to describe the results and their relation to operating conditions at HOSC.

5.1 Theoretical Performance Bounds

In Chapter 4, several statistics were discussed as being important indicators of network performance. These were identified as S, U, and D. S defined as throughput, is the rate of data flow on the network. U or utilization, is normalized throughput, this being the fraction of network capacity being utilized, and finally, D or delay, is the delay between the time an adapter requests the trunk and the time at which it completes a transmission. In order for the experimental values to have meaning, some theoretical determination of these statistics is needed so that comparisons may be made. In order to determine theoretically these values, the network factors which most affect HSLN performance must be known. The factors which affect

this performance, independent of the attached equipment are:

- **Bandwidth**
- **Propagation delay**
- **Number of bits per frame**
- **Local network protocol**
- **Offered load**
- **Number of stations [STALL84]**

The first three of these factors determine a parameter a, from which the network utilization U, and throughput S may be determined.

Since a local area network is distinguished from long haul networks and multiprocessor systems by the bandwidth (B) and distances (d) involved. The product of these two terms (B * d) can be used to characterize the network. The transmission length of the medium may be determined by dividing this quantity by the propagation velocity (V) of the medium. This quantity may be calculated as follows:

$$\text{Transmission Length} = \frac{B * d}{V}$$

Network Systems Corporation suggests a value of 40 percent of the speed of light be used for propagation velocity when attempting to establish network performance. Using the value of 50 Megabits per second for B, assuming d to be 1000 feet, and using a

value of 1.2 x 108 meters/sec for propagation velocity, the theoretical transmission length of a 1000 foot HYPERchannel network is 127 bits.

The ratio of transmission bit length, to the length of a typical frame (L) is the dimensionless quantity a. Thus,

$$a = \frac{\text{transmission length of trunk}}{\text{typical frame length (bits)}}$$

or

$$a = \frac{Bd}{VL}$$

Since d/V is the worst case propagation time on the medium it may be replaced with the HYPERchannel Fixed Delay. Recalling from Chapter 4 that Fixed Delay may be calculated as

Fixed Delay = 4 nsec * (trunk length) + 2.08 usec,

the parameter a may be calculated as

$$a = \frac{B}{L} * (\text{Fixed Delay})$$

Thus for a HYPERchannel trunk of length 1000 feet, a is determined to be 0.0109. Typical values of a for a HSLN range from 0.01 to over 1. An intuitive discussion of the meaning of the parameter a is found in William Stalling's Local Networks. This parameter is used to

provide an upper bound on the utilization of the network, Stallings also points out that this hypothesis appears to be supported by experimental evidence; a part of this discussion follows:

> ....Consider a perfectly efficient access mechanism that allows only one transmission at a time. As soon as one transmission is over, another node begins transmitting. Furthermore, the transmission is pure data--no overhead bits.... what is the maximum possible utiiization of the network? It can be expressed as the ratio of total throughput of the system to the capacity or bandwidth:
>
> $$U = \frac{\text{throughput}}{B}$$
>
> $$= \frac{L/(\text{Propagation + transmission time})}{B}$$
>
> $$= \frac{L/(D/V + L/B)}{B}$$
>
> $$= \frac{1}{1 + a}$$
>
> So, utilization varies inversely with a [STALL84].

Table 5.1 shows an example using a to predict the theoretical bounds on a network similar to the system at HOSC. Unfortunately, this method does not account for the access protocol of the network. Furthermore, it assumes that the maximum propagation time is incurred on each transmission and that only one transmission may occur at a time. Despite these shortcomings, upper

Table 5.1 Theoretical Bounds on a HYPERchannel Trunk of Length 1000 Feet.

Fixed Delay = 6.08 usec

Bit Length = 127.0325 bits

$$\underline{a} = 0.0109$$

$$U = \frac{1}{1 + \underline{a}} = 0.9892$$

$$S = U * B = 49.46 \text{ Mbits/sec}$$

For an arbitrary period of time, t, the total bits transferred are:

Total Bits = S * t

for t = 40 seconds

Total Bits = 1978 Megabits

bounds on throughput and utilization may be calculated using these techniques.

Another derivation of utilization which takes into consideration the effects of the protocol on the network performance, based on the work of Robert Metcalfe and David Boggs [METC76] is considered. Their work is based on the Ethernet system which is a local area network operating under a CSMA/CD medium access protocol with characteristics similar to HYPERchannel.

Assume that there are N active nodes on the network, each having identical data generating characteristics. Furthermore, after a station has transmitted its packet, another station becomes ready to transmit its packet, so that N also represents the total offered load on the network. Stallings in his amplification of [METC76] notes that time on the network may be divided into two components: transmission interval (time required to transmit a complete packet over the medium) and a contention interval (sequence of slots with a collision or no transmission). Throughput then, becomes the ratio of time transmitting to the total time.

Recall that a may be written as

$$a = \frac{\text{propagation time}}{\text{transmission time}}$$

where the propagation time is the worst case value of time to transmit a frame over the medium, and transmission time is the time required to get a complete frame onto the medium. If transmission time is normalized to one, then a becomes propagation time. The minimum length of the transmission slot must be chosen to allow for the detection of a collision and is equal to twice the end-to-end propagation time (1/2a).

Following Stalling's approach for the development of the equations describing throughput and utilization, the average length of the contention interval is determined by first computing A, the probability that exactly one station attempts a transmission in a slot and acquires access to the medium. This is the binomial probability that any one station attempts to transmit and the others do not,

$$A = \binom{N}{1} p^1 (1 - p)^{N-1}$$

$$= Np(1 - p)^{N-1}$$

which is the probability that the desired occurrence (acquisition of the trunk) will occur in N independent trials. Since maximum throughput and utilization are the quantities of interest, maximizing A will require that $p = 1/N$ so that,

$$A = (1 - 1/N)^{N-1}$$

The estimated mean length of contention interval, w, in slots is,

$$E[w] = \sum_{i=1}^{\infty} i * \Pr[\text{i slots in row with a collision or no transmission followed by a slot with no transmission}]$$

$$= \sum_{i=1}^{\infty} i(1 - A)^i A$$

which converges to,

$$E[w] = \frac{1 - A}{A} \quad \text{[STALL84]}.$$

The maximum value of utilization may now be determined, which is the length of a transmission interval as a proportion of a cycle consisting of a transmission and a contention interval

$$U = \frac{\text{transmission time}}{\text{total time}}$$

$$= \frac{\text{transmission time}}{\text{transmission time + contention time}}$$

$$= \frac{1/2a}{1/2a + (1 - A)/A}$$

$$= \frac{1}{1 + 2a\,\frac{1 - A}{A}} \quad .$$

For the 1000 foot HYPERchannel network previously described, assuming N = 4, U = 0.971 and S is 48.55 Megabits per second which corresponds nicely with the results obtained using only a (Table 5.1).

The calculation of the parameter D or message delay is not so obvious. It appears empirically, that D increases without bound as the network approaches saturation. As the number of nodes increases, so do the number of access requests, the number of collisions, the number of retransmissions and also the time delay a node experiences before access is obtained, so D, does appear to increase without bound.

5.2 Presentation of Simulation Results

The previous section presented analytical techniques useful for predicting the peak, or upper limits of a HYPERchannel network's performance. While this is useful for predicting top end performance, it does not provide any useful information about a particular configuration of the network.

The purpose of this simulation effort is to provide the engineers and scientists at HOSC with a software tool which will provide information on network performance in a variety of configurations. This section presents results from this model for the network descriptions shown in Figures 5.1 and 5.4. Detailed descriptions of these configurations, along

with simulation results are given in section 5.2.1 and the results for a dual trunk configuration are given in section 5.2.2.

5.2.1 Single Trunk HOSC Results

The single trunk configuration, as shown in Figure 5.1, and defined in terms of the model in Appendix IV, was considered. Table 5.3 contains a listing of the relative probabilities that a given transmitter will attempt to transmit to a given receiver. This is used by the simulation to establish the virtual connection between devices for communication. In order to gauge the performance of this configuration over a wide range of input loading, the topology of the network as defined was kept constant. The offered load to the network was varied by increasing the data source generation rates to the devices shown.

Table 5.2 illustrates the method for computing system performance measures from simulation statistics. Tables 5.4 and 5.5, contain data points calculated from simulation runs, with each data point representing a simulation run. These data points were plotted as shown in Figures 5.2 and 5.3.

Table 5.2 Calculation of Parameters using Simulation Statistics.

$$S \text{ (throughput)} = \frac{\text{Total Bits Transmitted}}{\text{Simulation Time}}$$

$$U \text{ (utilization)} = \frac{S}{\text{Bandwidth}}$$

$$= \frac{S \text{ (bits)}}{\text{50 Mbits per second}}$$

$$D \text{ (delay)} = \frac{\text{Avg. Message Delay Device 1} + \text{Avg. Message Delay Device 2} + \ldots \text{Avg. Message Delay Device n}}{n}$$

The above expression for delay is used in the case where transmitting devices have finite Average Message Delays. For the case where a transmitting Device attempts to transmit messages but has an Average Message Delay equal to zero (Average Message Delay = 0, and Abort count > 0) it is assumed, that Message Delay, or Delay, for the Network exceeded simulation time.

Offered Load is compiled by the simulation and output as one of its statistics.

Table 5.3 Relative Probabilities for Transmitter/ Receiver Pairs for Figure 5.1.

Transmitter/Receiver	Probability
111-311	0.7
111-321	0.3
211-431	1.0
221-421	1.0
311-211	0.01
311-221	0.01
321-211	0.01
321-221	0.01
331-421	1.0

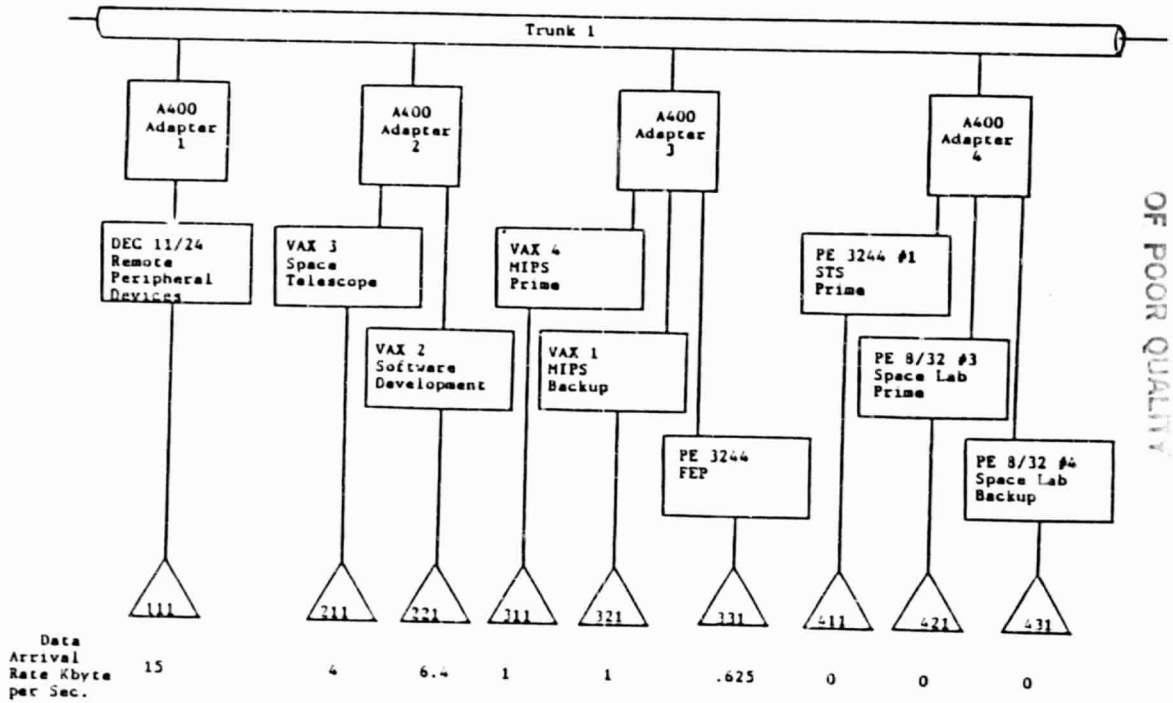


Figure 5.1 Single Trunk Simulation Set-Up.

Table 5.4 Data Points for Figure 5.2

S Mbytes/sec	U	G Mbytes	Average D secs
.0008092	.0001295	.0234998	.0009933
.0024680	.0003949	.0470083	.0021883
.0040020	.0006403	.0705255	.0019500
.0067190	.0010751	.1121140	.0017833
.0142275	.0022764	.2242640	.0016166
.0290956	.0046553	.4486710	.0015533
.0431299	.0069008	.6733410	.0016950
.0586719	.0093875	.8984520	.0016283
.0730755	.0116921	1.1224500	.0015116
.1421105	.0227377	2.2420000	.0018583
.2118972	.0339036	4.4840000	.0042466
.3202813	.0512450	6.7297000	.3875766
.5137168	.0821946	8.9692300	.0064750
.5548880	.0887821	11.2103000	.0042916
.9360150	.1497624	22.4200000	>40
1.5409865	.2465578	44.8401000	>40
2.3634718	.3781550	67.2603000	>40
2.3250275	.3720044	89.6800000	.4622500
3.4732500	.5557200	224.2000000	>40
3.6444500	.5831120	448.4000000	>40
4.5387500	.7261400	672.6000000	>40
4.6086250	.7373800	896.8000000	>40
4.6531500	.7445040	1121.0000000	>40
4.7345000	.7575000	2242.0000000	>40
4.3516500	.6962640	4340.0000000	>40
4.8016750	.7682680	6726.0000000	>40
4.8004000	.7680640	8968.0000000	>40
4.7978500	.7676560	11210.0000000	>40

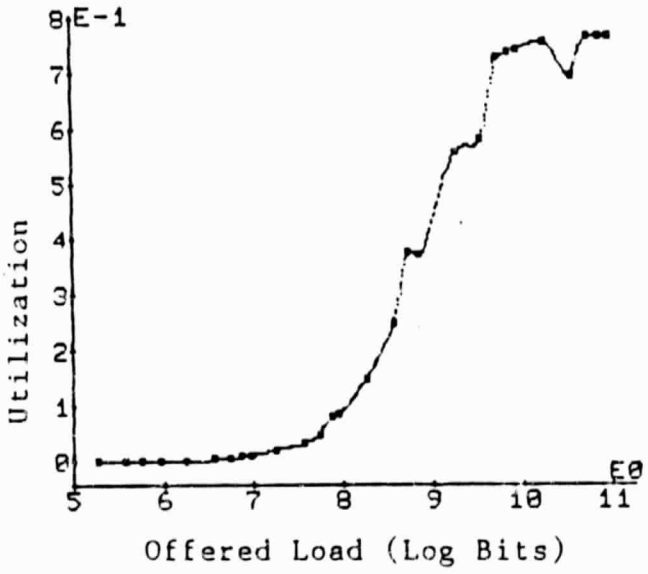
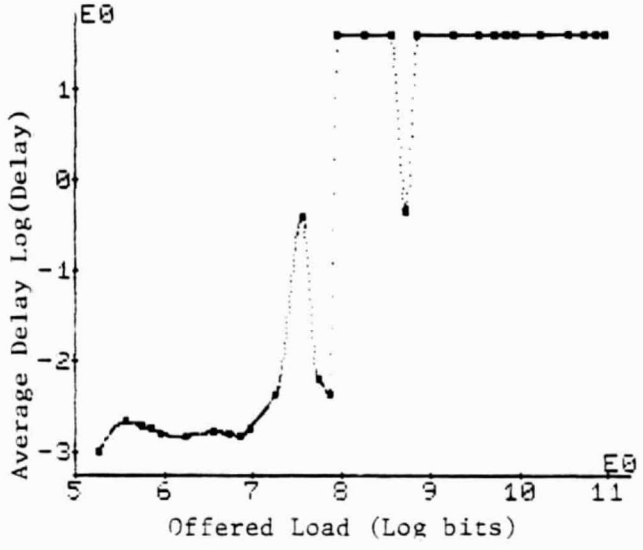


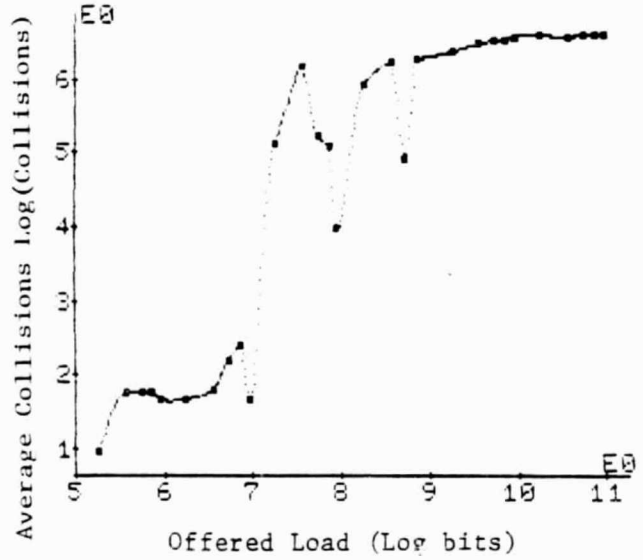
Figure 5.2 Utilization versus Offered Load
for Single Trunk Simulation.

Table 5.5 Data Points for Figure 5.3.

Control Bits Mbits	Data Bits Mbits	G Bits Mbits
.1005832	.177536	.1879984
.3076200	.520456	.3760664
.4885992	.854160	.5642040
.6720912	1.163416	.7210600
.7758192	1.374581	.8969120
1.6394960	2.914616	1.7941120
3.3552960	5.960952	3.5893680
5.0973920	8.719360	5.3867280
6.8325440	11.977040	7.1876160
8.4246400	14.989760	8.9796000
16.4381200	28.992240	17.9360000
27.1168800	40.690160	35.8720000
43.6347200	58.911760	53.8376000
62.5479200	101.864000	71.7538400
74.7621600	102.807200	89.6824000
134.9088000	164.616000	179.3600000
204.0576000	289.070400	358.7208000
347.8608000	408.488000	538.0824000
353.9296000	390.079200	717.4400000
539.0768000	572.364800	1793.6000000
685.7088000	480.517600	3587.2000000
582.9048000	869.376000	5380.8000000
564.0320000	910.728000	7174.4000000
564.3920000	924.616000	8968.0000000
543.6464000	971.392000	17936.0000000
624.7024000	767.824800	34720.0000000
527.8368000	1008.696000	53808.0000000
528.1136000	1008.008000	71744.0000000
528.7168000	1006.600000	89680.0000000

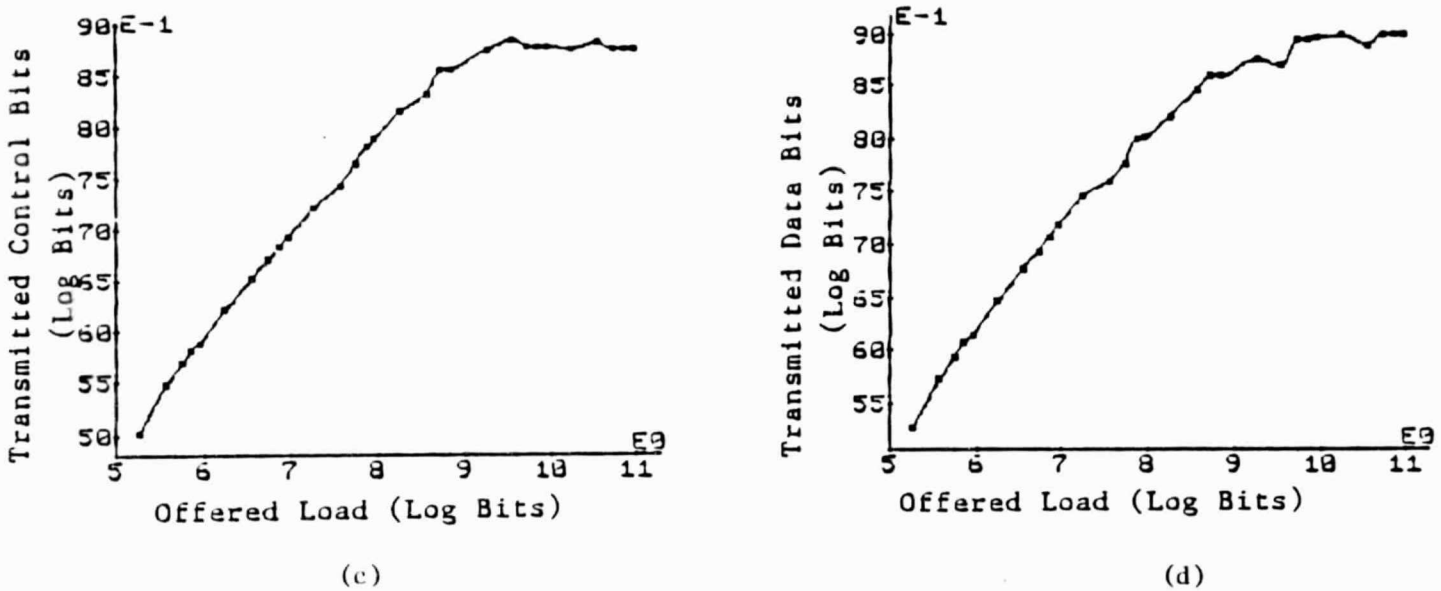


(a)



(b)

Figure 5.3 Plot of Transmitted Control and Data Bits versus Offered Load for Single Trunk Simulation.



(c) (d)

Figure 5.3 Plot of Transmitted Control and Data Bits versus Offered Load for Single Trunk Simulation.

The system configuration and data flow rates indicated in figure 5.1 are a typical loading of the main trunk of the HOSC system. The total offered load (G) for the system as depicted in figure 5.1 is computed as follows (G the offered load is the total number of data bits offered to the channel):

DEVICE	DATA ARRIVAL RATE
111	15 kbytes/sec = 120 kbps
211	4 kbytes/sec = 32 kbps
221	6.4 kbytes/sec = 51.2 kbps
311	1 kbytes/sec = 8 kbps
321	1 kbytes/sec = 8 kbps
331	.625 kbytes/sec = 5 kbps
411	0
421	0
431	0

TOTAL OFFERED LOAD = 224.2 kbps.

In a 40 second period a total of 8.976 Mbits or 6.953 (Log bits) would be offered for transmission across the HYPERchannel™ network. Referring to figure 5.2 the utilization of the HYPERchannel trunk for this offered load is approximately 2%. The throughput rate is thus (.02) * (50 Mbits per second) = 1 Mbps. Obviously the system is nowhere

near its upper limit in data throughput rate, trunk utilization and hence not in offered load.

The approximate number of total bits transmitted to send a 2 kbyte block of data may be estimated from figure 2.8 using Table 2.2 as a reference. From figure 2.8 it is apparent that 7 transmission frame format fields, 9 response frame fields, and 2 response frames with data format fields are transmitted to accomplish the transmission of 2 kbytes of data. This amounts to an absolute minimum transmission of 19600 bits total for 16000 bits of data or absolute minimum overhead of 22.5%. For longer data blocks the overhead is a smaller percentage and may be calculated by the relationship (assuming no collisions):

ABSOLUTE MINIMUM TOTAL BITS SENT

PER 2 KBYTES OF DATA = 19600 + (N-1)18600

where N = the number of 2 kbytes of data.

The reduction in transmission overhead for multiple transmission of data segements of 2 kbytes of data is not significant in number of bits transmitted but the timing and number of times the channel is free for contention is reduced by a significant amount. It will be observed from the

simulation runs that an average overhead of 50% for low offered load and 100% for high offered load is the case. This overhead increase is due to the non-integer multiples of 2 kbytes of data to be sent and to collisions.

Figure 5.3 depicts the average delay and number of collisions versus offered load. The average delay is an aggregate delay number and a large delay is indicative that at least one device is experiencing a large delay. Thus it may indicate that only one device is unable to transmit but all the remaining devices may be transmitting satisfactorily. However it is an indicator that things are begining to sour on the overall network performance.

Likewise the number of collisions is an aggregate sum and indicates the relative number of retry transmission attempts the system is experiencing. Since the HYPERchannel is not a pure contention system (such as ETHERNET) the increase of collisions does not necessarily indicate accompanying data loss or absolute inability of an adapter to communicate. Again it is an indicator that things are not so well on the overall network.

From the results of figure 5.3 it is apparent that as the offered load increases past about 7.5 (31.62 Mbits for 40 seconds or 790 kbits per second) the system is beginning to experience traffic congestion. The system's trunk utilization, figure 5.2, is seen to start a rise at this point (7.5) of offered load and it may be observed that at an offered load of about 9.5 (3162 Mbits for 40 seconds or 79 Mbits per second) the trunk utilization is entering saturation. At this point it is safe to say that some devices are just not able to transmit all their data. Somewhere in between these two offered loads is a point that would do well to be identified as the maximum useful offered load condition. Using the rule of thumb for normal maximum operating offered load for pure contention systems (ETHERNET is an example of a pure contention system) the maximum operating offered load would be set to approximately 1/2e or 18.35% of the channel bandwidth. For slotted contention systems this rule of thumb figure is approximately 1/e or 36.7%. The HYPERchannel is a hybrid priority/contention system with prioritized time slotting and it is reasonable to expect to load the system to the 36.7% trunk utilization. Inspecting

the graph of trunk utilization versus offered load, figure 5.2, it is observed that the offered load for a utilization figure of 36.7% is approximately 8.5 (log bits) or a 7.9 Mbps aggregrate rate (316 Mbits in a 40 second).

Inspection of the graphs for the transmitted control bits and the transmitted data bits versus offered load, figure 5.3, also illustrate that at 8.5 offered load the control bits and data bit curves start to enter saturation. Thus it is reasonable to state that the maximum operating offered load (in pure aggregrate rate data bits) should be 8.5. This is an aggregrate rate of offered data of 7.9 Mbps or 15.8% of the channel bandwidth of 50 Mbits. Remember that an absolute minimum of 22.5% overhead is needed at low collision rates and thus 7.9 Mbps of offered data plus 22.5% overhead is approximately 9.6775 Mbps or 19.35% of the channel bandwidth. (This assumes all data blocks are integral 2 Kbytes in size).

A further comment of the figure 5.3c and 5.3 d is in order. The offered data to the channel, G, (see Table 5.5) is only the actual data to be transmitted. The actual number of data and control bits required (as evidenced from the simulation

runs), which when added together equal the channel throughput S, will not equal G but be greater than G.

It should be observed that for low offered load the actual transmitted bits on the channel amounts to 2.54 times the offered load and at high offered load this figure decreases to about 2 times the offered load until saturation takes place. This apparent increase in efficiency is misleading since the prioritized time slotting mode of operation replaces the contention portion of the channel algorithm and stagnation of the throughput (U*B) takes place.

Thus the 7.9 Mbps offered aggregate load corresponds to about 16 Mbps to 18 Mbps actual channel offered total load or 36% of the channel bandwidth.

FROM THIS DISCUSSION AND THE SIMULATION RESULTS COUPLED WITH THE ANALYTICAL RESULTS

IT IS RECOMMENDED THAT THE TOTAL AGGREGATE OFFERED DATA TO THE HYPERchannel SHOULD BE HELD TO 8 Mbps OR LESS. IN FACT DUE TO THE UNCERTAINTY OF ESTIMATING AGGREGATE DATA RATES A MAXIMUM OF 4 Mbps WOULD SEEM PRUDENT.

5.2.2 Dual Trunk HOSC Results

The dual trunk configuration as shown in Figure 5.4 and detailed in Appendix V, was considered. Once again, the relative probabilities for transmitter/
receiver pairs in this configuration are contained in Table 5.6. In order to compare the effect of the dual trunk configuration with the previous results, the configuration of trunk 2 in Figure 5.4 was defined as the single trunk configuration previously considered. The topology of this configuration was held constant and the offered load varied, the resulting data points are contained in Tables 5.7, 5.8, and 5.9. These data points were plotted as shown in Figures 5.5, 5.6, and 5.7.

In a secenario that has been judged to be a normal environment for the HOSC dual trunk system, the transmissions from trunk 1 to trunk 2 have aggregate data rate of 10 kbytes/sec. The device on trunk 1 always transmitts across the trunks to trunk 2. The cross traffic from trunk 2 to trunk 1 has a lower data rate and a lower probability of request for transmission. Thus the majority of

Table 5.6 Relative Probabilities for Transmitter/ Receiver Pairs for Figure 5.4.

Transmitter/Receiver	Probability
111-521	1.0
211-411	0.7
211-421	0.3
311-531	1.0
321-521	1.0
321-111	0.3
411-311	0.01
411-321	0.01
421-311	0.01
421-321	0.01
431-521	1.0

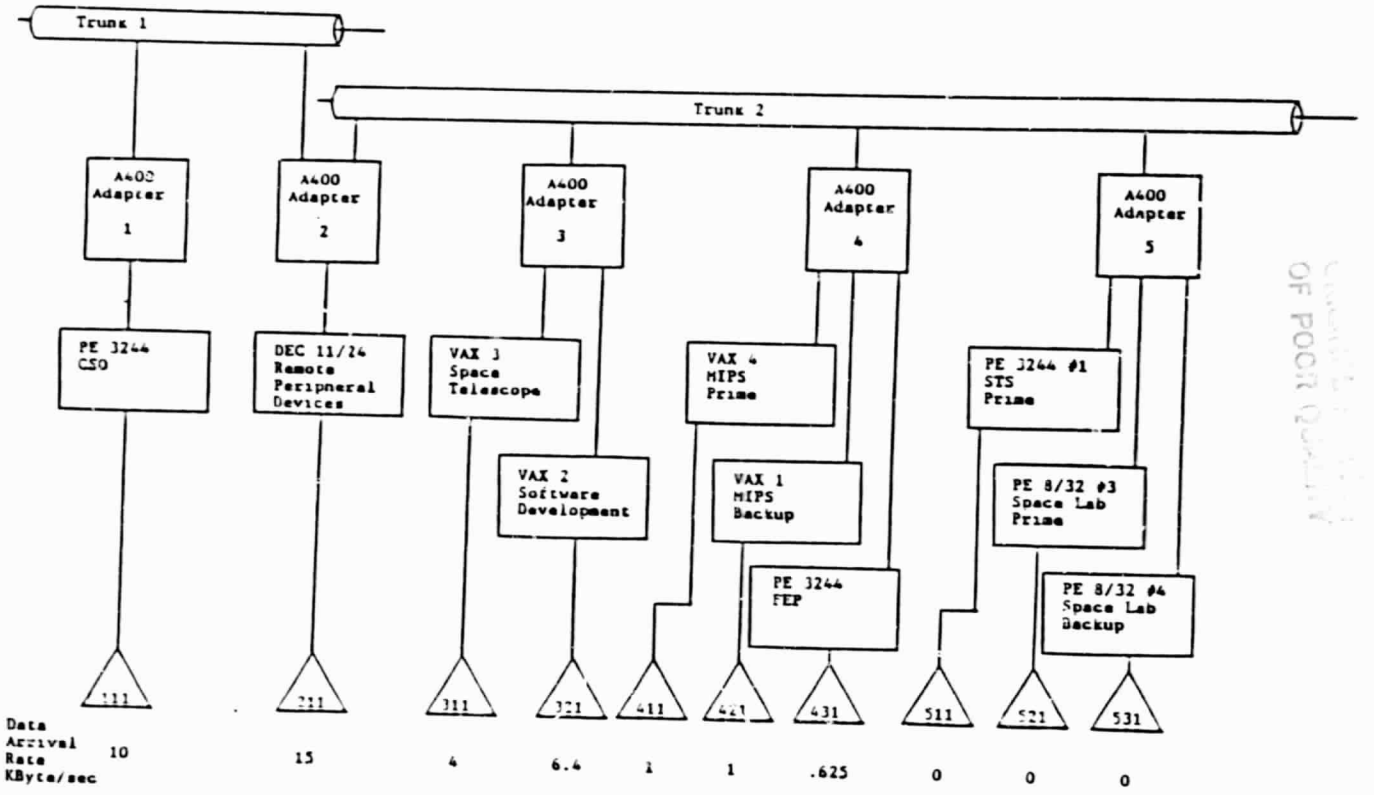


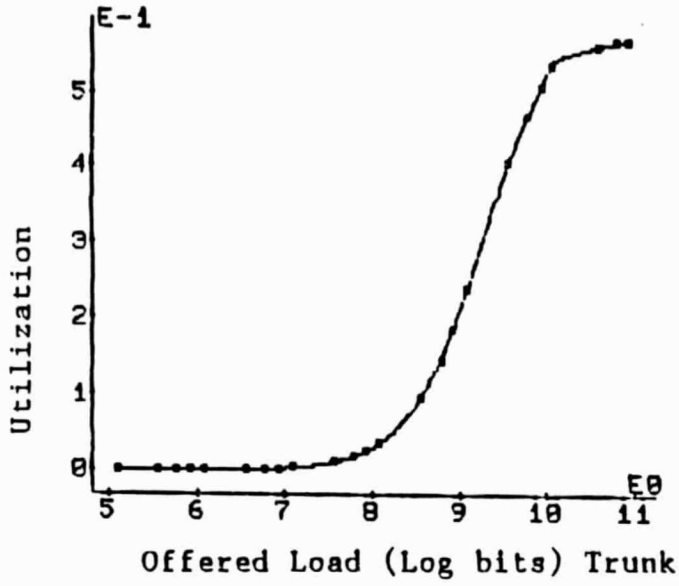
Figure 5.4 Dual Trunk Simulation Set-Up.

Table 5.7 Data Points for Figure 5.5 (Trunk 1)

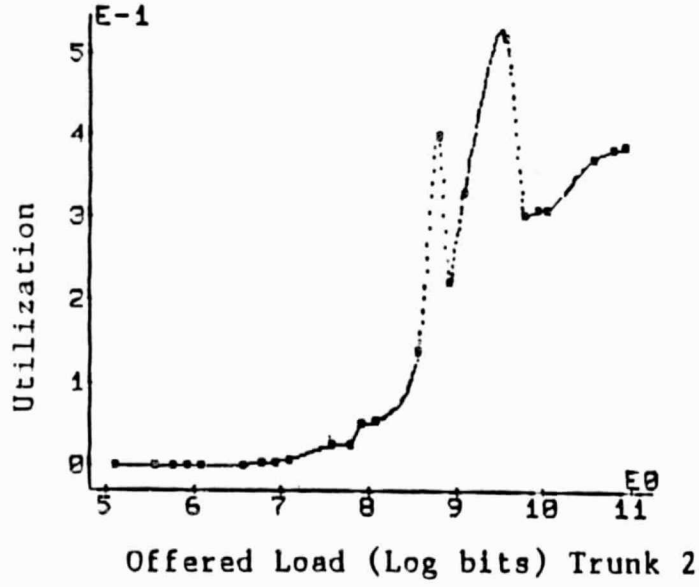
S Mbytes/sec	U	G Mbytes	Average D secs
.00017973	.000028758	.0155758	.0006650
.00065895	.000105433	.0467330	.0018928
.00116560	.000186500	.0760532	.0018800
.00161704	.000258726	.1090710	.0015842
.00239248	.000382797	.1521190	.0014942
.00711533	.001138454	.4564050	.0013442
.01189340	.001902940	.7610820	.0013142
.01665555	.002664880	1.0658600	.0013000
.02376770	.003802830	1.5233900	.0013000
.07053307	.011285292	4.5644800	.0013228
.11621025	.018593600	7.6050000	.0064371
.16076790	.025722800	10.6496000	.0014928
.22569486	.036111111	15.2116000	.0067786
.60678250	.097085200	45.6300000	.0288557
.91607500	.146572000	76.0500000	.0977300
1.17207750	.187532400	106.4700000	>40
1.50327750	.240524400	152.1000000	.7125557
2.52247200	.403590000	456.3000000	>40
2.93269750	.469231600	760.5000000	>40
3.15442500	.504700000	1064.7000000	>40
3.34350000	.534960000	1377.0000000	>40
3.49356500	.558970000	4563.0000000	>40
3.52475250	.563960400	7605.0000000	>40
3.54453500	.567125600	10647.0000000	>40

Table 5.8 Data Points for Figure 5.5 (Trunk 2).

S Mbytes/sec	U	G Mbyte	Average D sec
.00017973	.000028758	.0155758	.0006650
.00142330	.000227728	.0467330	.0018928
.00204550	.000327280	.0760532	.0018800
.00321635	.000514616	.1090710	.0015842
.00478900	.000766240	.1521190	.0014942
.01557770	.002492400	.4564050	.0013442
.02641426	.001902940	.7610820	.0013142
.03715009	.005944016	1.0658600	.0013000
.05333650	.008533800	1.5233900	.0013000
.16056906	.025691050	4.5644800	.0013228
.16651150	.026641800	7.6050000	.0064371
.32703950	.052326322	10.6496000	.0014928
.34178971	.054686000	15.2116000	.0067786
.86553250	.138485200	45.6300000	.0288557
2.49451500	.399122400	76.0500000	.0977300
1.38427372	.221483800	106.4700000	>40
2.05963550	.329541600	152.1000000	.7125557
3.24608750	.519374000	456.3000000	>40
1.88915240	.302260000	760.5000000	>40
1.92437440	.307899000	1064.7000000	>40
1.93925400	.310280000	1377.0000000	>40
2.31418420	.370269000	4563.0000000	>40
2.38785975	.382057500	7605.0000000	>40
2.40758250	.385200000	10647.0000000	>40

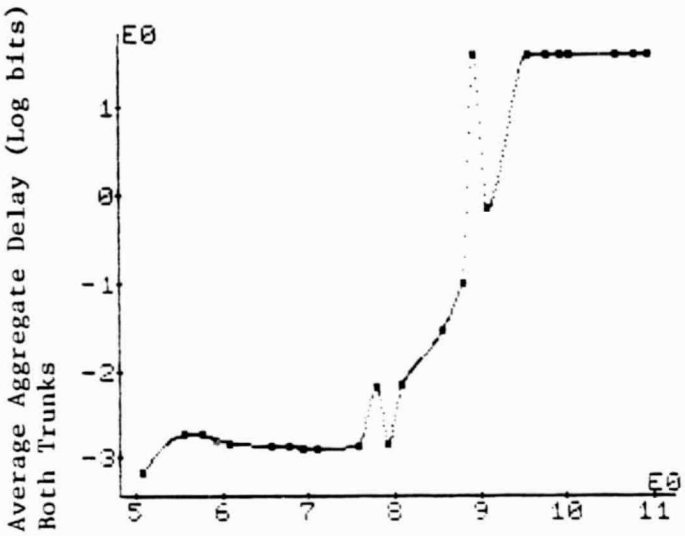


(a)

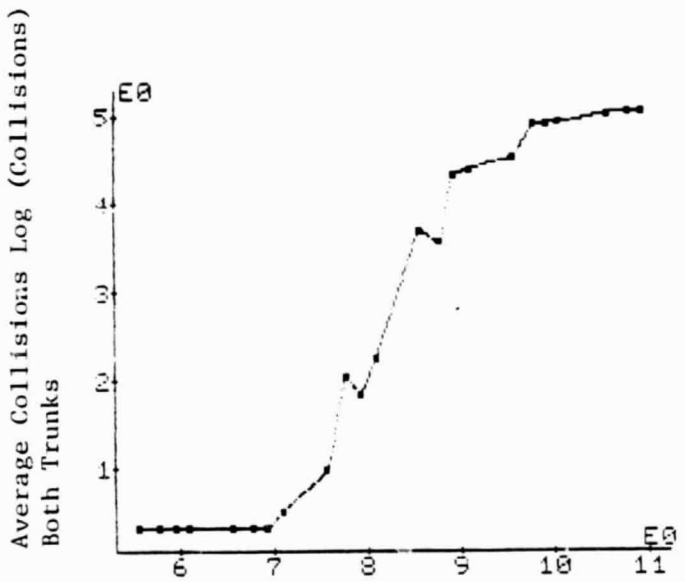


(b)

Figure 5.5 Plot of Utilization versus Offered Load for Dual Trunk Simulation.



(c)



(d)

Figure 5.5 Plot of Utilization versus Offered Load for Dual Trunk Simulation.

Table 5.9 Data Points for Figures 5.6 and 5.7.

Trunk 1		Trunk 2		
Control Mbits	Data Mbits	Control Mbits	Data Mbits	G Mbits
.327870	.026112	.327870	.026112	.12460
.120219	.095744	.229786	.236680	.37386
.207651	.165376	.339076	.315528	.60842
.295083	.235008	.528275	.526096	.87256
.426231	.339456	.761190	.771480	1.21695
1.267768	1.009664	2.416856	2.569176	3.65124
2.120224	1.688576	4.083456	4.375576	6.08865
2.972688	2.367488	5.728200	6.172760	8.52688
4.240456	3.377152	8.220800	8.873680	12.18712
12.568320	10.009600	24.684128	26.717520	36.51584
20.699520	16.485360	44.027040	9.256640	60.84000
28.644880	22.813200	59.468240	45.209520	85.19680
40.207760	32.022000	93.209600	16.174320	121.69280
108.088000	86.082400	193.222400	83.748000	365.04000
163.210400	129.933600	411.237600	38.700720	608.40000
208.792000	166.272800	440.861600	2.105992	851.76000
274.172000	206.876800	631.889600	27.193760	1216.80000
449.335200	357.856000	896.320000	142.428000	3650.40000
522.411200	416.052000	607.649600	.032768	6084.00000
561.916800	447.499200	615.689600	.110208	8517.60000
595.596800	474.324800	620.444800	.116488	11016.00000
740.745600	457.195200	717.393600	23.145360	36504.00000
6.3.976000	453.948000	733.200800	30.914320	60840.00000
679.876000	454.375200	732.420800	38.005600	85176.00000

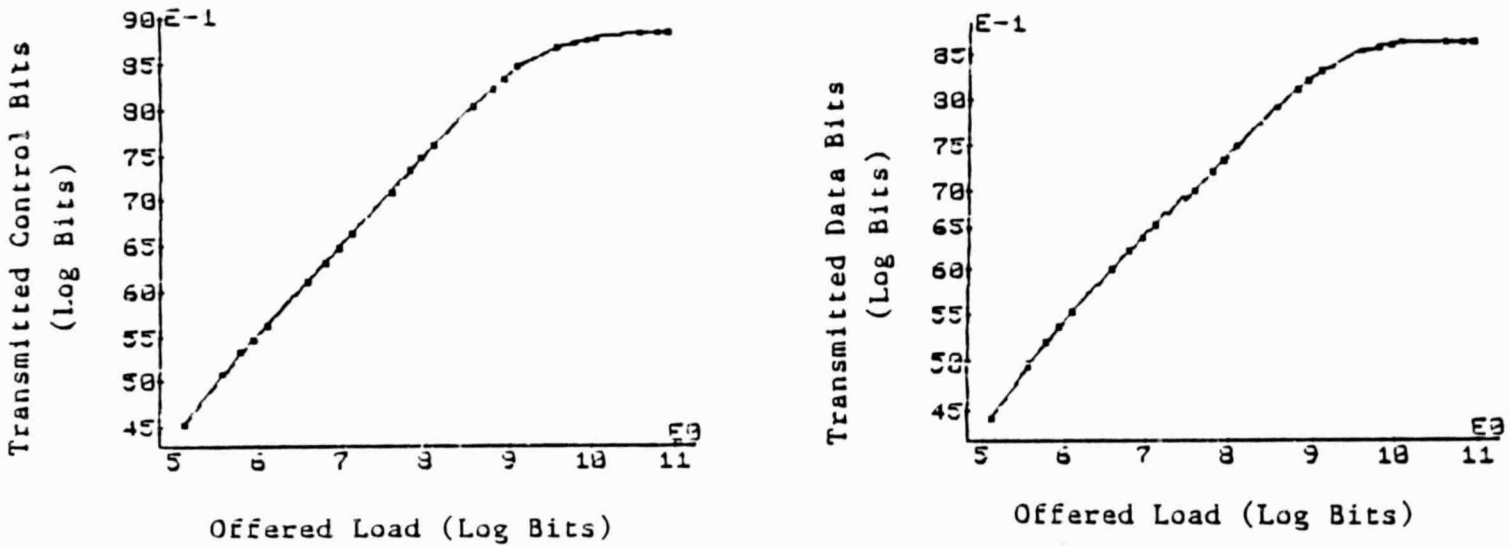


Figure 5.6 Plot of Transmitted Control and Data Bits versus Offered Load for Dual Trunk (Trunk 1) Simulation.

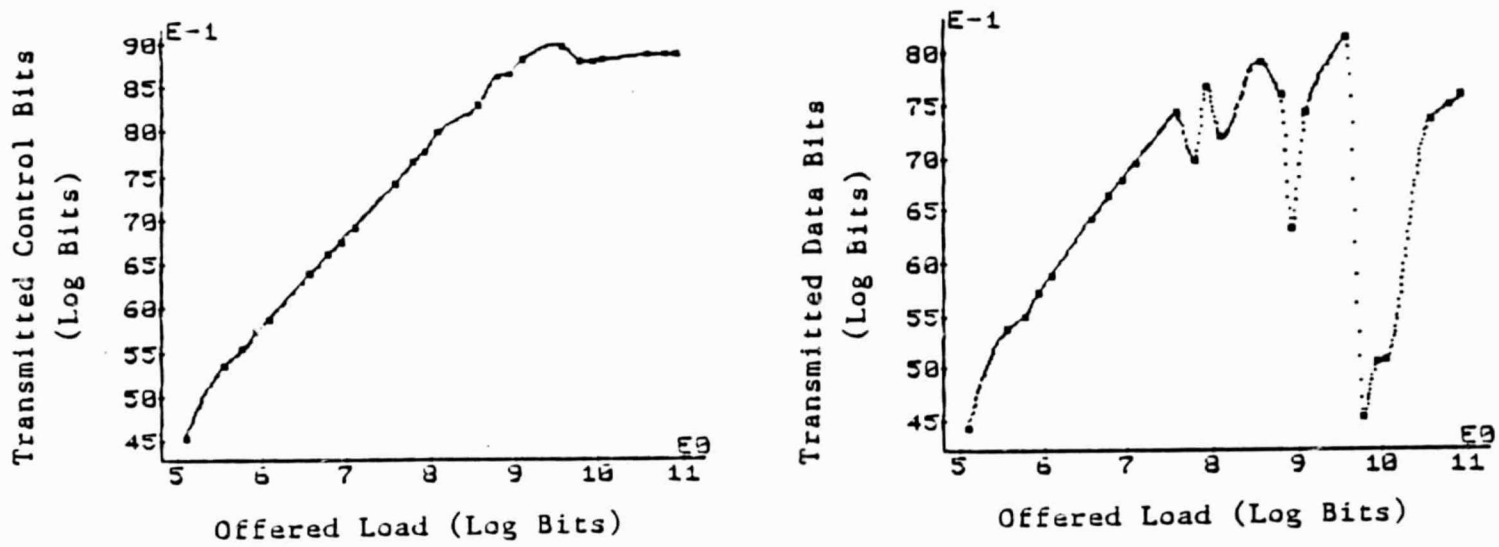


Figure 5.7 Plot of Transmitted Control and Data Bits versus Offered Load for Dual Trunk (Trunk 2) Simulation.

the cross trunk transmissions occur from trunk 1 to trunk 2. This is envisioned as the worst case condition for cross trunk traffic since the destination trunk is the busiest trunk.

Looking at the curves from figures 5.5, 5.6 and 5.7 it may be observed that for trunk 1 the operational parameters under increasing offered load are very similar to that for the single trunk secenario, the exception being the lowered trunk utilization before saturation is encountered. Again the maximum offered load operationg point is seen to occur for about 8.5 offered load and hence as before the trunk 1 aggregate offered load should be held to 8 Mbps or less.

The situation is not as good for trunk 2. The cross trunk traffic hampers the normal traffic flow for trunk 2 and it is apparent from the simulation curves that the maximum aggregate offered data load should be held to less than 8 or 2.5 Mbps. This is considerably less than that for the single trunk simulation and clearly demonstrates the effect of heavy cross trunk communication requirements.

From figure 5.4 the aggregate offered data load of the two trunks may be calculated. This

was judged to represent a typical operating load for the dual trunk HOSC system.

DEVICE	DATA ARRIVAL RATE
111	10 kbytes/sec = 80 kbps
TRUNK 1 TOTAL OFFERED LOAD =	80 kbps
211	15 kbytes/sec = 120 kbps
311	4 kbytes/sec = 32 kbps
321	6.4 kbytes/sec = 51.2 kbps
411	1 kbytes/sec = 8 kbps
421	1 kbytes/sec = 8 kbps
431	1 kbytes/sec = 8 kbps
511	0
521	0
531	0
TRUNK 2 TOTAL OFFERED LOAD =	224.2 kbps.

The aggregate offered data load for trunk 1 is 80 kbps (6.505 log bits) and for trunk 2 is 224.2 kbps (6.952 log bits). These offered aggregate data rates are less than the desired maximum by a considerable amount and room for growth exists. There is an allowable factor of 100 for trunk 1 and an allowable factor of 10 for trunk 2 growth rates. Keeping in mind that this aggregate data load for

trunk 2 is the same as for the single trunk simulation configuration of figure 5.1 it is easily discerned that the cross trunk traffic while amounting to a 35% increase in trunk 2 traffic lowers the maximum operating offered load potential of trunk 2 by a factor of almost 4 (down from 8 Mbps to 2.5 Mbps). Furthermore the curve of figure 5.5 illustrates that the utilization of trunk 1 saturates at 55% as opposed to the 75% point for the single trunk case.

FOR DUAL CONFIGURATIONS IT SHOULD BE NOTED THAT THE MAXIMUM AGGREGATE OPERATING LOAD POTENTIAL OF THE TRUNKS IS LOWERED BY A CONSIDERABLE AMOUNT. THIS ARISES FROM THE NECESSITY OF DUAL RESERVATION OF BOTH TRUNK ADAPTERS. THE CROSS TRUNK TRAFFIC LOWERS THE MAXIMUM OPERATING UTILIZATION POTENTIAL TO APPROXIMATELY 75% OF THAT FOR THE SINGLE TRUNK.

FURTHERMORE CARE SHOULD BE TAKEN IN ASSIGNING DEVICES TO THE TRUNK ON WHICH THEY ARE MOST ACTIVE.

5.3 Analysis and Conclusions

The goal of this work was to develop a software tool, with particular emphasis on NSC's HYPERchannel network, to aid the engineers and scientists at the HOSC, to perform system analysis. This study began with a description of the HYPERchannel and the various protocols implemented by the network. The relationship of these protocols to system performance was discussed. The system components which make up the HOSC were examined, these components are examined for information purposes, in that the attributes of these devices were used to define the test configurations presented in section 5.2. The configurations chosen for testing by the simulation, are based on the system shown in Figure 3.1. The results of this testing are presented in section 5.2 of this work.

The data, which appear in Section 5.2, are plotted in all cases versus the Log (base 10) of offered load, where offered load is expressed in bits. So that an offered load of 7 on any plot is the inverse Log (base 10) of 7 which is equal to 1 x 107 bits. In Figures (5.2 and 5.5) utilization is plotted versus offered load. Utilization is a dimensionless figure ranging from 0 to 1. In Figures (5.3, 5.6, and 5.7) the Log (base 10) of transmitted control and data bits are plotted versus offered load. As before a value of 7 on

the offered load axis represents 1 x 10^7 bits. Whether this is data or control bits, is indicated on the axis of its respective plot. Finally, the simulation time used, 40 seconds, was the same for all simulation runs.

The results obtained from the single trunk configuration of the model are found in Figures 5.2 and 5.3. The results as plotted in Figure 5.2 and tabulated in Table 5.3, compare favorably with results obtained by other researchers [FRAN82], [FRAN84], and [WATS80]. While numerical results differ due to different topologies being modeled, overall performance of the network is essentially the same. The overall performance of this network, and that reported elsewhere, appears to approach some relatively constant level (saturation) as offered load increases. A slightly different presentation of these results are plotted in Figure 5.3 and tabulated in Table 5.5. This data is presented as transmitted control and data bits versus offered load. These plots represent the actual amount of data and control bits being transmitted by the network versus offered load. Control bits are those bits transmitted by an adapter to perform network functions and are considered overhead loading. Data bits transmitted, are those bits offered to the network by attached devices for transfer over the network. Offered load represents the total amount of data

presented to the network for transmission over the network. Thus, as offered load increases, the amount of data bits transmitted should increase. Since network services are requested, network functions increase, thereby increasing the amount of network messages relating to these functions. This will cause a corresponding increase in the amount of control bits being transmitted over the network. The plots in Figure 5.3, both exhibit nearly linear properties until a certain point (offered load approximately equal to 9) is reached, after which the data and control bits transmitted remain nearly constant.

An offered load of 9 on these plots, represents an aggregate offered load of 3.125 Mbytes/sec being presented to the network for transmission. This is equivalent to an off network source requesting network services every 0.04 usec. According to published figures [FRAN82], [FRAN84], and [WATS82], an adapter requesting the trunk, transmitting its data, and then releasing itself and its receiving adapter will require a minimum of 840 usec to send one message-with-data sequence containing one 2 Kbyte data block. If the off network sources' data rates are such that network services are requested faster than the network can process these requests, then data loss will occur. At this point the network is entering saturation. This is

reflected in the plots of data bits transmitted versus offered load. After offered load exceeds the point where offered load is approximately 9, any further increase in the offered load does not produce a corresponding increase in the amount of data and control bits transmitted. This indicates that the network is saturated and can not transmit any greater amount of data or control bits.

The normal offered load to the network, based on observations of the system described in Chapter 3, and detailed in Appendix IV, appears to be approximately 6.95 (Offered Load). This figure results from summation of the data rates applied to the network, multiplied by simulation time, and then performing a Log (base 10) operation on the result. Thus this network is, according to simulation results, operating below its potential by a factor of approximately 100. This indicates that expansion of this configuration would be possible to some extent.

In order to examine the effects of a second trunk added to a network, the configuration shown in Figure 5.4, and detailed in Appendix V, was modeled. This configuration is similar to the single trunk configuration in that the topology of trunk 2 (Figure 5.4) is the same as that shown in Figure 5.1. In order to observe the effects on trunk 2 due to another trunk

having access to it, the probability of a transmission to trunk 2 from trunk 1 was set higher than the probability of the reverse operation. It should be apparent, that a dual trunk network in which each trunk has need of only sporadic use of the other, would have noticeably better performance than the configuration presented in this work. The results of modeling this configuration are plotted in Figures 5.5 through 5.7, and are tabulated in Tables 5.7 through 5.9. Examination of these results shows that trunk 1 is relatively unaffected, although utilization is lower. Utilization in this case is approximately constant at a value of 0.55 for increasing offered load. This figure is approximately 25% less than the comparable value for a single trunk network, this is due largely to the topology of the network. On trunk 1 there are two adapters, which communicate exclusively with each other. There is a larger number of adapters on trunk 2; which affects the common adapter to both trunks, in that, multi-trunk operation is slower. This slowness of the multi-trunk operation may be attributed to the common adapter having to compete for a busier trunk to complete its' transmission. Thus, the common adapter will be reserved for longer periods of time, directly affecting throughput, and consequently utilization, on both trunks. Performance on trunk 1 of this topology,

with the exception of lower utilization, is roughly the same as that of a single trunk network, as expected. However, the results for trunk 2 are drastically reduced from that of a single trunk configuration. An examination of Figures 5.2 and 5.5, show that utilization of the trunk is very nearly the same for an offered load of approximately 5 to a load of very nearly 8. After an offered load of 8 is reached, the dual trunk configuration's utilization is drastically reduced. In Figure 5.6, the control and data bits transmitted over trunk 1 are plotted. These curves exhibit behavior similar to that of the single trunk model discussed previously. This is not the case for the control and data bits transmitted over trunk 2 (Figure 5.7). Both of these plots exhibit linear behavior to a certain extent, but while the control bits transmitted are only slightly affected, the transmitted data is drastically altered at the same point that utilization breaks down. Based on the description of this network contained in Appendix V, the normal offered load is approximately 7.08. As shown in the plotted data, this topology has very limited range in regard to offered load, which makes it vulnerable to performance degradation due to load variation.

Thus, a single trunk system exhibits greater potential for expansion than the dual trunk topology modeled. Rather than allow a dual trunk system of this nature, it would be better (Performance related) to place the additional device seeking communication with another trunk, at a high rate, physically on the trunk it requests the most.

The validity of this simulation is based on the understanding of the system and the structure of the model. While it would be better to establish validity by comparison with actual configurations and their measured performance, the feasibility of establishing a test bed for this purpose is remote, due largely to the intensive mission requirements at the HOSC.

Finally, the author believes that the stated research goals have been reached. However, in this model, as in all things there is considerable room for improvement. Future expansion of the model might be made, to include intra-adapter message transfers, and more than one adapter to more than one trunk. This would serve to lend more accuracy and sophistication to the present model. The current model appears to possess reasonable accuracy and validity, with the stated conditions under which it was developed.

APPENDIX I

Software Source Listing

```
1PASCAL COMPILER( SAU ) *** RELEASE VOS 2.2 ***  1 APR 85 10:56:01 ***

     1    {*************************************************************************
     2     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     3     * *                                                                     * *
     4     *                                                                       *
     5     * *   TITLE:                                                          * *
     6     *       A SOFTWARE MODEL OF NETWORK SYSTEMS CORPORATION'S HYPERCHANNEL   *
     7     * *       NETWORK.  A MODEL OF THE HUNTSVILLE OPERATIONS SUPPORT       * *
     8     *           CENTER.                                                     *
     9     * *   AUTHOR:                                                         * *
    10     *       DENNIS L. MASSEY                                              *
    11     * *                                                                   * *
    12     *       MISSISSIPPI STATE UNIVERSITY                                  *
    13     * *   MAY, 1985                                                       * *
    14     *                                                                       *
    15     * *                                                                   * *
    16     *                                                                       *
    17     * *   ABSTRACT:                                                       * *
    18     *                                                                       *
    19     * *         NETWORK SYSTEMS CORPORATION'S HYPERCHANNEL NETWORK IS A    * *
    20     *     BASEBAND, 50 MEGABIT PER SECOND, HIGH SPEED LOCAL AREA NETWORK.   *
    21     * *   HYPERCHANNEL UTILIZES A CARRIER SENSE MULTIPLE ACCESS (CSMA)    * *
    22     *     DATA TRUNK ACCESS PROTOCOL, WITH PRIORITIZED STAGGERED DELAYS.    *
    23     * *   THIS NETWORK HAS THE CAPABILITY OF INTERCONNECTING VARIOUS      * *
    24     *     COMPUTER RESOURCES OF DIFFERING MANUFACTURE, INTO A SINGLE        *
    25     * *   EFFICIENT COMPUTING FACILITY.                                   * *
    26     *           THIS NETWORK IS CURRENTLY IN USE AT THE HUNTSVILLE OPERATIONS *
    27     * *   SUPPORT CENTER (HOSC). HOSC AS A DISTRIBUTED COMPUTING SYSTEM,  * *
    28     *     IS RESPONSIBLE FOR DATA AQUISITION AND ANALYSIS DURING NATIONAL   *
    29     * *   AERONAUTICS AND SPACE ADMINISTRATION'S (NASA) SPACE SHUTTLE     * *
    30     *     OPERATIONS. HOSC ALSO PROVIDES COMPUTING SERVICES FOR MARSHALL    *
    31     * *   SPACE FLIGHT CENTER'S (MFSC) NON-MISSION ACTIVITIES. AS MISSION * *
    32     *     AND NON-MISSION ACTIVITIES CHANGE, SO DO THE SUPPORT FUNCTIONS OF *
    33     * *   HOSC CHANGE, DEMONSTRATING THE NEED FOR SOME METHOD OF SIMULATING * *
    34     *     ACTIVITY AT HOSC IN VARIOUS CONFIGURATIONS.                       *
    35     * *         THE SIMULATION DEVELOPED IN THIS WORK PRIMARILY MODELS NSC'S * *
    36     *     HYPERCHANNEL NETWORK, SINCE THIS NETWORK BASICALLY DETERMINES     *
    37     * *   HOSC'S EFFICIENCY IN ACCOMPLISHING ITS MISSION. THE MODEL SIMULATES* *
    38     *     THE ACTIVITY OF A STEADY-STATE NETWORK, REPORTING STATISTICS SUCH  *
    39     * *   AS, TRANSMITTED BITS, COLLISION STATISTICS, FRAME SEQUENCES TRANS- * *
    40     *     MITTED, AND AVERAGE MESSAGE DELAY. THESE STATISTICS MAY BE USED TO *
    41     * *   EVALUATE SUCH PERFORMANCE INDICATORS AS THROUGHPUT, UTILIZATION,  * *
    42     *     AND DELAY. THUS, THE OVERALL PERFORMANCE OF THE NETWORK MAY BE     *
    43     * *   EVALUATED, AS WELL AS PREDICTING POSSIBLE OVERLOAD CONDITIONS.   * *
    44     *                                                                       *
    45     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    46     *************************************************************************}
    47
    48   PROGRAM SIM1 (INPUT, OUTPUT, DFILE (LFN=17), AUXOUT (LFN=16), OUT (LFN=18));
    49
    50    {*********************************************************}
    51    {            D A T A    D E C L A R A T I O N S           }
    52    {*********************************************************}
    53
```

```
54      CONST
55        MAXNUMSOURCES = 3;
56        MAXNUMADAPTERS = 9;
57        PROPAGATION = 1.32E-09;     (* PROP. DELAY  NS/FT *)
58        TRUNKRATE = 50.0E+06; (* TRUNK TX RATE IN BITS/ SECS. *)
59        ONE = 1;
60        ZERO = 0;
61        MAXNUMTRUNKS = 2; (* MAXNUMBER TRUNKS IN NETWORK *)
62
63      TYPE
64          POINTER = ^LIST;
65            LIST  =  RECORD
66                     ID : INTEGER;
67                     PROB : REAL;
68                     NEXT : POINTER
69                     END;  (* LINKED RECORD *)
70
71
72
73
74
75
76        SOURCEDESCRIPTIONS = RECORD
77                               ID: PACKED ARRAY [1..24] OF CHAR;
78                               DNUM   : INTEGER; (* DEVICE NUMBER *)
79                               DEST         : INTEGER;  (* ID OF DESTINATION
80                                                          WHEN TX *)
81                               IDEST : INTEGER; (* INTERMEDIATE DESTINATION WHEN
82                                                  TRUNK-TRUNK TX *)
83                               FROMTX : INTEGER; (* TRANSMITTER WHEN RECEIVING *)
84                               BUFFERSIZE: REAL;  (*  NUMBER OF BITS ACCUMULATED
85                                                    FROM A OFF-NET SOURCE BEFORE
86                                                    THE DEVICE REQUESTS A TRUNK
87                                                    TRANSMISSION *)
88                               DISTFROMA1: REAL; [ DISTANCE FROM ADAPTER 1 ]
89                               DISTFRTA1   : REAL; (* USED IN DUAL TRUNK, FOR
90                                                    2ND TRUNK      *)
91                               TFERRATE: REAL; (* DEVICE TO ADAPTER I/O RATE *)
92
93                               NEXTTX: REAL;(* TIME UNTIL NEXT TRANSMISSION *)
94
95                               TXINTRVL: REAL; [ TIME BETWEEN TX REQUESTS BASED ON
96                                                 AGGREGATE SOURCE TRANS. RATES FROM
97                                                 OFFNET SOURCES ]
98                               SEQCT : INTEGER;
99                               WAITCT: INTEGER; [ DEVICE WAITS TALLY ]
100                              COLLCT: INTEGER; [ DEVICE COLLISION TALLY ]
101                              RSEQCT : INTEGER; (* DEVICE RECEPTION TALLY *)
102                              ABORTCT  : INTEGER; [ DEVICE ABORT  TALLY *)
103                              BUSYTIME : REAL;
104                              WAITTIME: REAL; [ TIME DEVICE SPENT IN WAIT DELAYS ]
105                              COLLTIME: REAL; [ TIME DEVICE SPENT IN COLL. WAITS ]
106                              GENRATE: REAL; [ AVG NUMBER OF BYTES/SEC ]
107                              M1 : BOOLEAN;
108                              M2 : BOOLEAN;
```

```
109                          M3 : BOOLEAN;     (* MESSAGE FLAGS *)
110                          M4 : BOOLEAN;
111                          M5 : BOOLEAN;
112                          M6 : BOOLEAN;
113                          M8 : BOOLEAN; (* USED FOR 2 TRUNK OPS *)
114                          BLKSENT : REAL;
115                          TBLK : REAL ;
116                          DATABLKCT : REAL;
117                          INITIALTX : REAL;      (* INITIAL TX TIME *)
118                          FINALTX   : REAL;      (* FINAL TX TIME *)
119                          MESSDELAY : REAL;  (* SUM OF FINALTX - INITIALTX *)
120                          FIRST  : POINTER;
121                        END;
122
123        DEVICERECORDS = RECORD
124                          ID: PACKED ARRAY [1..24] OF CHAR;
125                          OPEN: BOOLEAN; [ TRUE IF DEVICE ATTACHED TO PORT ]
126                          NUMOFSOURCES: INTEGER;
127                          SOURCE: ARRAY [1..MAXNUMSOURCES] OF SOURCEDESCRIPTIONS;
128                          TFERRATE: REAL; [ I/O PORT TRANSFER RATE FROM DEV TO ADAPT
129                                            IN BYTES/SEC ]
130                          LOADTIME: REAL; [ INVERSE OF TFER_RATE = TIME FOR DEV TO
131                                            LOAD ADAPT BUFFER ]
132                          SFLAG: ARRAY [1..3] OF BOOLEAN;
133
134
135
136
137
138
139
140
141
142                      END;
143
144        ADAPTERRECORDS = RECORD
145                           DEVICE: ARRAY [1..4] OF DEVICERECORDS;
146                           DISTFROMA1: REAL; [ DISTANCE IN FEET FROM ADAPTER 1 ]
147                           DISTFRTA1   : REAL ;
148                           PRI1DELAY    : REAL; [ NUMBER OF SECONDS OF ASSIGNED WAIT
149                                                  TIME IN EVENT OF WAIT DURING
150                                                  TRANSMISSION ]
151                           END1DELAY : REAL;
152                           FIX1DELAY : REAL; (*END TO END TX TIME ON TRUNK 1 *)
153                           TOT1DELAY : REAL;  (* TOTAL DELAY PERIOD FOR TRUNK 1 *)
154                           DELAY : REAL ;   (* DELAY USED IN RESERVATION SCHEME *)
155                           RETRYCT : INTEGER; (* BASED ON ADAPTER UNIT NUMBER
156                                                 RANGES FROM 0-65, UNIQUE FOR EACH
157                                                 ADAPTER (RESERVATION SCHEME )*)
158                           RETRY : INTEGER;
159                           PRI2DELAY : REAL; (* PRIORITY DELAY FOR TRUNK 2       *)
160                           END2DELAY : REAL; (* END  DELAY FOR TRUNK2       *)
161                           FIX2DELAY : REAL; (* FIXED DELAY FOR TRUNK2      *)
162                           TOT2DELAY : REAL; (* TOTAL DELAY FOR TRUNK2      *)
163                           FINALDEST : INTEGER; (* USED BY COMMON ADAPTER *)
```

```
164                         P1 : INTEGER; (* INTEGER VALUE OF PRIORITY DELAY ON
165                                          TRUNK 1 *)
166                         P2 : INTEGER; (* INTEGER VALUE OF PRIORITY DELAY ON
167                                          TRUNK 2 *)
168                         DFLAG : ARRAY [1..4] OF BOOLEAN;
169
170
171
172                       END;
173
174
175       [TEXT = FILE OF CHAR;]
176
177       MODETYPE = (INTERACTIVE, FILEINPUT);
178
179       CLOCKCOND = (CASE1,CASE2,CASE3,CASE4,CASE5,CASE6,CASE7);
180
181     VAR
182       NUMOFADAPTERS: INTEGER; [ NUMBER OF ADAPTERS IN NET ]
183       ADAPTER: ARRAY [1..MAXNUMADAPTERS] OF ADAPTERRECORDS;
184       NUMOFTRUNKS: INTEGER;
185       TRUNKLENGTH: ARRAY [1..MAXNUMTRUNKS] OF REAL; [ DIST IN FEET ]
186                 TOTALABORTS, MAXSEQTX,                                COLLTALLY,
187       SEQTALLY, SEQ1TALLY, SEQ2TALLY, TRK2ATTEMPT, TRK2TALLY, TA, TD, TS,
188       TOTALATTEMPTS,
189        COMMAD     , I, J, K, DNUM:INTEGER;
190       OFFLOAD,CONT1BITS, DATA1BITS, CONT2BITS, DATA2BITS, WAITTALLY,
191       TR1ACTIVE, TR2ACTIVE, PRIDELAY, TOTALDELAY, ENDDELAY, FIXEDDELAY,
192       MAXTIME, BITSTX, CARUP, CARDWN,TIMEDIF,                  T1, T2, T3, T4,
193        CURRENTTIME, [ SYS CLOCK RECORDING TOTAL ELAPSED TIME ]
194        TIMEACTIVE, LASTTRUNKTX: REAL; [ TIME OF LAST SUCCESSFUL TRUNK TRANS. ]
195
196
197       U, IU: INTEGER; [ SEED FOR RANDOM NUMBER GENERATOR ]
198       HEADING: PACKED ARRAY [1..50] OF CHAR; [ PROGRAM HEADING ]
199       RESP: PACKED ARRAY [1..80] OF CHAR;
200       DFILE, AUXOUT, OUT: TEXT;
201       INPUTMODE: MODETYPE;
202       CONDITION: ARRAY [CLOCKCOND] OF CLOCKCOND;
203       COND1: CLOCKCOND;
204      TMITTER, RCVR: SOURCEDESCRIPTIONS;
205       PRINTALL, TRK1, RESERVED : BOOLEAN;
206       LCT, PCT, JK: INTEGER;
207       TRUNKS : BOOLEAN; (* TRUE FOR DUAL TRUNK *)
208       C        , CURRENT, PTR : POINTER;
209       [*********************************************************]
210       [*********************************************************]
211       [       P R O C E D U R E     D E F I N I T I O N S       ]
212       [*********************************************************]
213
214       [*************************************************************]
215       [                                                             ]
216
217
218     PROCEDURE INITIALIZE;
```

```
219       [       PERFORMS BEGINNING OF RUN VARIABLE INITIALIZATIONS AND ]
220       [       SETS UP SIMULATION RUN PARAMETERS.                     ]
221       [                                                              ]
222       [       INVOKED BY: PROGRAM MOD3C                              ]
223        [       CALLS TO  : NO OTHER ROUTINES
224        (*                                                              ]
225       [       GLOBAL VARIABLES AFFECTED:U,WAIT_TALLY,COLLISION_TALLY,]
226       [           TOTAL_ATTEMPTS,LCT,PCT,TIME_ACTIVE,BITS_TX,        ]
227       [           MAX_TIME,PRINT_ALL,INPUT_MODE.                     ]
228       [                                                              ]
229       [**************************************************************]
230
231         VAR
232           COND1: CLOCKCOND;
233         L,I,J,K: INTEGER;
234        BEGIN [ PROCEDURE INITIALIZE ]
235
236          [ NEW RUN INITIALIZATIONS ]
237
238
239      FOR I := 1 TO MAXNUMADAPTERS DO
240         FOR J := 1 TO 4 DO
241          WITH ADAPTER[I] DO
242           BEGIN
243          FOR L := 1 TO 4 DO
244            DFLAG[L] := FALSE;
245            FOR K := 1 TO MAXNUMSOURCES DO
246               WITH ADAPTER[I],DEVICE[J],SOURCE[K] DO
247                WITH ADAPTER[I].DEVICE[J] DO
248                  BEGIN
249                    SEQCT := 0;
250                    WAITCT := 0;
251                    COLLCT := 0;
252                    RSEQCT := 0;
253                    WAITTIME := 0.0;
254                    COLLTIME := 0.0;
255                    GENRATE := 0.0;
256                    NEW(FIRST);
257                    SFLAG[K]  := FALSE;
258                    M1 := FALSE;
259                    M2 := FALSE;
260                    M3 := FALSE;
261                    M4 := FALSE;
262                    M5 := FALSE;
263                    M6 := FALSE;
264                    M8 := FALSE;
265                    BLKSENT := 0.0;
266                    IDEST := 0;
267                    DEST := 0 ;
268                    FROMTX := 0;
269                    TBLK := 0.0;
270                    BUSYTIME := 0.0;
271                    ABORTCT  := 0;
272                    RETRY    := 0;
273                    MESSDELAY := 0;
```

```
274                 END;
275               END;
276
277          FOR COND1 := CASE1  TO CASE7 DO CONDITION[COND1] := COND1;
278          PRINTALL := TRUE;
279          DNUM:= 0;
280          U :=  16227; { U IS SEED FOR RNG }
281          WAITTALLY := 0.0;
282          COLLTALLY := 0;
283          TOTALABORTS   := 0;
284          LCT := 0;
285          PCT := 0;
286          TIMEACTIVE := 0.0;
287          BITSTX := 0.0;
288          TRUNKS := FALSE;
289          MAXTIME := 00.0;
290          MAXSEQTX := 0000000;
291          PRIDELAY := 0.0;
292          TOTALDELAY := 0.0;
293          FIXEDDELAY := 0.0;
294          ENDDELAY   := 0.0;
295          TR1ACTIVE := 0.0;
296          TR2ACTIVE := 0.0;
297          TRK1 := FALSE;
298          RESERVED := FALSE;
299          SEQTALLY := 0;
300          SEQ1TALLY := 0;
301          SEQ2TALLY := 0;
302          TOTALATTEMPTS := 0;
303          TRK2ATTEMPTS := 0;
304          TRK2TALLY := 0;
305          CONT1BITS := 0.0;
306          DATA1BITS := 0.0;
307          CONT2BITS := 0.0;
308          DATA2BITS := 0.0;
309          T1 := 0.0;
310          T2 := 0.0;
311          T3 := 0.0;
312          T4 := 0.0;
313          OFFLOAD := 0.0;
314
315
316          REWRITE(OUT);
317
318          WRITELN;
319
320          WRITELN(' IS THERE MORE THAN ONE TRUNK ? Y)ES OR N)O');
321          REPEAT READ(RESP[1]) UNTIL RESP[1] <> ' '; READLN;
322            IF (RESP[1] = 'Y') OR (RESP[1] = 'Y') THEN
323                  TRUNKS := TRUE
324               ELSE
325                  TRUNKS := FALSE;
326
327
328
```

```
329
330         WRITELN;
331         WRITELN( 'IDENTIFICATION HEADING FOR RUN:');
332         READLN( HEADING);
333         WRITELN;
334
335         WRITELN;
336         WRITELN( 'MAXIMUM RUN TIME IN SECS?');
337         READLN( MAXTIME);
338         WRITELN;
339
340         WRITELN( 'MAXIMUM SUCCESSFUL SEQUENCE TRANSMISSIONS ? ');
341         READLN( MAXSEQTX );
342         WRITELN;
343
344         WRITELN;
345         WRITELN( 'SEED FOR RANDOM NUMBER GENERATOR?');
346         READLN( U);
347         IU := U;
348         WRITELN;
349
350
351         WRITELN;
352
353         IF TRUNKS THEN
354            BEGIN
355             WRITELN('WAS THE LAST RUN A SINGLE TRUNK CONFIGURATION Y)ES OR N)O');
356               REPEAT READ(RESP[1]) UNTIL RESP[1] <> ' '; READLN;
357                    IF (RESP[1] = 'Y') OR (RESP[1] = 'Y') THEN
358                         BEGIN
359                         INPUTMODE := INTERACTIVE;
360                         WRITELN(' INPUTMODE IS INTERACTIVE ');
361                         END
362                     ELSE
363                         BEGIN
364
365         WRITELN( 'INPUT MODE:  I)NTERACTIVE OR F)ILE ');
366         REPEAT READ( RESP[1]) UNTIL RESP[1] <> ' '; READLN;
367         IF (RESP[1] = 'I') OR (RESP[1] = 'I') THEN
368           INPUTMODE := INTERACTIVE
369         ELSE
370           INPUTMODE := FILEINPUT;
371       END;
372     END;
373      IF NOT(TRUNKS) THEN
374           BEGIN
375             WRITELN(' WAS THE LAST RUN A DUAL TRUNK CONFIGURATION Y)ES OR N)O');
376              REPEAT READ(RESP[1]) UNTIL RESP[1] <> ' '; READLN;
377                   IF (RESP[1] = 'Y') OR (RESP[1] = 'Y') THEN
378                     BEGIN
379                         INPUTMODE := INTERACTIVE;
380                         WRITELN(' INPUTMODE IS INTERACTIVE ');
381                      END
382         ELSE
383             BEGIN
```

```
384                 WRITELN( 'INPUT MODE : I)NTERACTIVE OR F)ILE ? ');
385                   REPEAT READ(RESP[1]) UNTIL RESP[1] <> ' ';READLN;
386                     IF (RESP[1] = 'I') OR (RESP[1] ='I') THEN
387                           INPUTMODE := INTERACTIVE
388                       ELSE
389                           INPUTMODE := FILEINPUT;
390                   END;
391               END;
392           PAGE(OUTPUT);
393
394        END; [ PROCEDURE INITIALIZE ]
395        [*************************************************]
396        [                                                 ]
397
398
399       PROCEDURE CHARACTERIZENETWORK;
400        [        PROMPTS FOR NETWORK DESCRIPTIVE INFO.    ]
401        [                                                 ]
402        [        INVOKED BY: PROGRAM SIM1                 ]
403        [        CALLS TO  : TX INTERVAL                  ]
404        [                    REWRITE DFILE                ]
405        [                                                 ]
406        [                                                 ]
407        [        GLOBAL VARIABLES AFFECTED: ADAPTER[I]    ]
408        [                                                 ]
409        [*************************************************]
410
411         VAR
412           I, J, K, L, A: INTEGER;
413           TEMP, TEMP1, B: REAL;
414           SORT : ARRAY [1..MAXNUMADAPTERS] OF INTEGER;
415           SORT1 : ARRAY [1..MAXNUMADAPTERS] OF INTEGER;
416
417           [*********************************************************]
418           [                                                         ]
419
420
421         PROCEDURE REWRITEDFILE;
422          [       REWRITES  THE AUXILLIARY INPUT FILE               ]
423          [*********************************************************]
424
425           VAR
426             I, J, K, L: INTEGER; [ LOOP VARIABLES ]
427
428           BEGIN [ PROCEDURE REWRITE ]
429             (*OPEN(DFILE,'DESCRIP',OLD);*)
430             REWRITE(DFILE);
431
432             WRITELN(DFILE, NUMOFTRUNKS);
433             FOR I := 1 TO NUMOFTRUNKS DO WRITELN(DFILE, TRUNKLENGTH[I]);
434
435             WRITELN(DFILE, NUMOFADAPTERS);
436             WRITELN(DFILE, COMMAD );
437             FOR I := 1 TO COMMAD   DO
438               BEGIN
```

```
439             WITH ADAPTER[I] DO
440               BEGIN
441               WRITELN(DFILE, DISTFROMA1);
442               WRITELN(DFILE, PRI1DELAY    );
443               WRITELN(DFILE, END1DELAY );
444               WRITELN(DFILE, RETRYCT );
445              IF ( TRUNKS ) AND ( I = COMMAD ) THEN
446                 BEGIN
447                   WRITELN(DFILE, DISTFRTA1);
448                   WRITELN(DFILE, PRI2DELAY);
449                   WRITELN(DFILE, END2DELAY);
450                 END;
451
452               {   BEGIN INDIVIDUAL DEVICE DESCRIPTIONS   }
453
454               FOR J := 1 TO 4 DO
455                 BEGIN
456                 WITH ADAPTER[I],DEVICE[J] DO
457                   BEGIN
458                   WRITELN(DFILE, OPEN);
459                   IF NOT (OPEN) THEN
460                     BEGIN
461
462                     WRITELN(DFILE, ID);
463
464                     WRITELN(DFILE, TFERRATE);
465
466                     WRITELN(DFILE, NUMOFSOURCES);
467                     FOR K := 1 TO NUMOFSOURCES DO
468                       WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
469                         BEGIN
470                         WRITELN(DFILE, DNUM  );
471                         WRITELN(DFILE, ID);
472                         WRITELN(DFILE, GENRATE);
473                         WRITELN(DFILE, BUFFERSIZE);
474                         WRITELN(DFILE, DATABLKCT );
475                          WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
476                          BEGIN
477                              C := FIRST;
478                               WHILE C <> NIL DO
479                                    BEGIN
480                                      WRITELN(DFILE,C^.ID);
481                                      WRITELN(DFILE, C^.PROB);
482                                      C := C^.NEXT;
483                                    END;
484                                WRITELN(DFILE, 0 );
485                          END;
486
487                         END; { FOR K:=1 TO NUM OF SOURCES }
488                     END; { IF NOT(OPEN) }
489                   END; { WITH ADAPTER().DEVICE() }
490                 END; { FOR J=1 TO 4 }
491               END; { ?WWITH ADAPTER() }
492             END; { FOR I=1 TO NUM OF ADAPTERS }
493
```

```
494         IF TRUNKS THEN
495            BEGIN
496               FOR I := ( COMMAD + 1 ) TO NUMOFADAPTERS DO
497                BEGIN
498                  WITH ADAPTER[I] DO
499                    BEGIN
500                     WRITELN(DFILE, DISTFRTA1 );
501                     WRITELN(DFILE, PRI2DELAY );
502                     WRITELN(DFILE, END2DELAY );
503                     WRITELN(DFILE, RETRYCT );
504                      FOR J := 1 TO 4 DO
505                       BEGIN
506                        WITH ADAPTER[I].DEVICE[J] DO
507                         BEGIN
508                           WRITELN(DFILE, OPEN);
509                             IF NOT(OPEN) THEN
510                               BEGIN
511                                 WRITELN(DFILE, ID );
512                                 WRITELN(DFILE, TFERRATE );
513                                 WRITELN(DFILE, NUMOFSOURCES );
514                      FOR K := 1 TO NUMOFSOURCES DO
515                        WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
516                          BEGIN
517                            WRITELN(DFILE, DNUM );
518                            WRITELN(DFILE, ID   );
519                            WRITELN(DFILE, GENRATE );
520                            WRITELN(DFILE, BUFFERSIZE );
521                            WRITELN(DFILE, DATABLKCT );
522                            WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
523                             BEGIN
524                               C := FIRST;
525                                 WHILE C <> NIL DO
526                                   BEGIN
527                                     WRITELN(DFILE, C^.ID);
528                                     WRITELN(DFILE, C^.PROB);
529                                     C := C^.NEXT;
530                                   END;
531                            WRITELN (DFILE, 0 );
532                             END;
533                          END;
534                       END;
535                    END;
536                END;
537             END;
538          END;
539       END;
540
541          END; [ PROCEDURE REWRITE ]
542
543
544
545
546
547
548
```

```
549
550
551
552
553
554
555
556
557     BEGIN [ PROCEDURE CHARACTERIZE NETWORK ]
558
559       [       BEGIN DESCRIPTION OF NETWORK BY DESCRIBING ADAPTERS     ]
560
561       CASE INPUTMODE OF
562
563         INTERACTIVE:
564           BEGIN
565
566           WRITELN('*****************************************************');
567           WRITELN('         ***************************************');
568           WRITELN;
569           WRITELN('                  DESCRIPTION OF NETWORK');
570           WRITELN('           DEVICE BY DEVICE -- ADAPTER BY ADAPTER');
571           WRITELN;
572
573           IF NOT( TRUNKS ) THEN
574               NUMOFTRUNKS := 1
575           ELSE
576               NUMOFTRUNKS := 2;
577
578           WRITELN('LENGTH OF TRUNKS IN FEET (USED FOR END DELAY)');
579           FOR I := 1 TO NUMOFTRUNKS DO
580             BEGIN
581             WRITELN( 'TRUNK ', I: 1, ':');
582             READLN( TRUNKLENGTH[I]);
583             END;
584
585           WRITELN;
586           WRITELN( 'NUMBER OF ADAPTERS IN NETWORK:');
587           READLN( NUMOFADAPTERS);
588
589           IF TRUNKS THEN
590             BEGIN
591               WRITELN('WHICH ADAPTER IS COMMON TO BOTH TRUNKS ?');
592               READLN ( COMMAD );
593             END;
594           IF NOT(TRUNKS) THEN
595              COMMAD := NUMOFADAPTERS;
596
597
598
599
600
601
602           FOR I := 1 TO COMMAD DO
603             WITH ADAPTER[I] DO
```

```
604                 BEGIN
605                 WRITELN;
606                 WRITELN('ADAPTER #', I: 2, ':');
607                 WRITELN;
608                 WRITELN( ' ': 2,
609                   'DISTANCE IN FT FROM PRIORITY 1 ADAPTER ON TRUNK 1 : ');
610                 READLN( DISTFROMA1);
611
612                 WRITELN;
613                 WRITELN( ' ': 2,
614                     'ADAPTER PRIORITY: GIVE IN INTEGER FORM,IE. N= 1,2...');
615                 WRITELN(' ': 2,'THIS IS USED TO CALCULATE PRIORITY DELAY ');
616                 READLN( P1);
617                 FIX1DELAY  := TRUNKLENGTH[1] * 4.0E-09 + 2.08E-06;
618                 WRITELN;
619                 WRITELN;
620                 WRITELN( ' ':2,
621                     ' ADAPTER RETRY COUNT : THIS MUST BE A UNIQUE NUMBER');
622                 WRITELN(' ':25,'FOR EACH ADAPTER IN THE RANGE 0 TO 64.');
623                 READLN ( RETRYCT );
624                 WRITELN;
625
626                  IF ( TRUNKS ) AND ( I = COMMAD ) THEN
627                     BEGIN
628                      WRITELN;
629                      WRITELN(' ':2,
630                      'DISTANCE IN FT FROM PRIORITY 1 ADAPTER ON TRUNK 2 : ');
631                      READLN( DISTFRTA1 );
632                      WRITELN;
633                      WRITELN(' ':2,
634                        'ADAPTER PRIORITY: GIVE IN INTEGER FORM, IE. 1,2,...');
635                      WRITELN(' ':2,'THIS IS USED TO CALCULATE PRIORITY DELAY');
636                      READLN ( P2 );
637                      WRITELN;
638                      FIX2DELAY := TRUNKLENGTH[2] * 4.0E-09 + 2.08E-06;
639                      WRITELN(' ':2, 'THE ADAPTER COMMON TO BOTH TRUNKS');
640                      WRITELN(' ':2, 'MUST HAVE AT LEAST ONE ATTACHED ');
641                      WRITELN(' ':2, 'DEVICE WITH AN ACTIVE DATA SOURCE');
642                     END;
643
644                 {   BEGIN INDIVIDUAL DEVICE DESCRIPTIONS   }
645
646                 FOR J := 1 TO 4 DO
647                   BEGIN
648                   WITH ADAPTER[I],DEVICE[J] DO
649                     BEGIN
650                     OPEN := TRUE;
651                     WRITELN;
652                     WRITELN(' ': 2, 'ADAPTER ', I: 2, ' DEVICE ', J: 2,
653                             ' DESCRIPTION');
654                     WRITELN( ' ': 4,'DEVICE STATUS: O)PEN OR C)LOSED ');
655                     REPEAT READ( RESP[1]) UNTIL RESP[1] <> ' '; READLN;
656                     IF (RESP[1] = 'C') OR (RESP[1] = 'C') THEN OPEN := FALSE;
657                     IF NOT (OPEN) THEN
658                       BEGIN
```

```
659
660                     WRITELN( ' ': 4, 'DEVICE ID (<=24 CHAR)');
661                     READLN( ID);
662
663                     WRITELN( ' ': 4,
664                           'DEVICE TO ADAPTER I/O RATE (BYTES/S):');
665                     READLN( TFERRATE);
666                     TFERRATE := TFERRATE * 8;
667                     LOADTIME := 1.0 / TFERRATE;
668
669                     WRITELN( ' ': 4,
670                           'NUMBER OF DATA SOURCES IN DEVICE ', J: 1);
671                     READLN( NUMOFSOURCES);
672
673                     FOR K := 1 TO NUMOFSOURCES DO
674                       WITH ADAPTER[I],DEVICE[J],SOURCE[K] DO
675                         BEGIN
676                         DISTFROMA1 := ADAPTER[I].DISTFROMA1;
677                         IF ( I = COMMAD ) AND (TRUNKS) THEN
678                         DISTFRTA1  := ADAPTER[I].DISTFRTA1;
679                         WRITELN;
680                         WRITELN( ' ': 5, 'DATA SOURCE ', K:1 , ' ID');
681                         READLN( ID );
682
683                         DNUM := I * 100 + J * 10 + K;
684                         WRITELN;
685                         WRITELN( ' ': 5, 'SOURCE[', K: 1,
686                                 '] DATA GEN RATE (BYTES/S)');
687                         READLN( GENRATE);
688                         GENRATE := GENRATE * 8;
689                         WRITELN( ' ': 5, 'BUFFER SIZE IN BYTES:');
690                         READLN( BUFFERSIZE);
691                         BUFFERSIZE := BUFFERSIZE * 8;
692                         DATABLKCT := BUFFERSIZE / 16384;
693                         TFERRATE := ADAPTER[I].DEVICE[J].TFERRATE;
694                         OFFLOAD := OFFLOAD + GENRATE;
695                         IF GENRATE > 0.0 THEN
696                           TXINTRVL := BUFFERSIZE / GENRATE
697                         ELSE
698                           TXINTRVL := 1E7;
699
700                         NEXTTX := TXINTRVL;
701
702                     WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
703                      BEGIN
704                        WRITELN(' ':2,
705                           'A LIST OF POSSIBLE RECEIVERS AND THE PROBABILITY');
706                        WRITELN(' ':2,'OF ADAPTER[',I:1,'], DEVICE[',J:1,
707                            '], SOURCE[',K:2,'] TRANSMITTING ');
708                        WRITELN(' ':2,'TO THEM WILL NOW BE MADE.');
709                        WRITELN;
710                        WRITELN(' ':2,
711                           'ID OF THE RECEIVER WILL BE A 3 DIGIT CODE BASED');
712                        WRITELN(' ':2,
713                           'ON THE ADAPTER#, DEVICE#, AND SOURCE#. ');
```

```
                    WRITELN(' ':2,
                        'THE PROBABILITY MUST BE GIVEN AS A REAL NUMBER');
                    WRITELN(' ':2,
                        'LESS THAN OR EQUAL TO 1. ENTER  0  FOR ID TO ');
                    WRITELN(' ':2,'COMPLETE THE LIST.');

                     CURRENT := FIRST;
                    (*NEW(CURRENT); FIRST := CURRENT;*)
                     WRITELN('RECEIVER ID CODE ? ( 0 TO END LIST)');
                     READLN( A );
                     WRITELN;
                      IF ( A = 0 ) THEN
                       BEGIN
                         CURRENT := NIL;
                         C := CURRENT;
                         FIRST := CURRENT;
                       END
                      ELSE
                       BEGIN
                         WRITELN('PROBABILITY ?');
                         READLN ( B );
                       CURRENT^.ID := A;
                       CURRENT^.PROB := B;
                         C := CURRENT;
                       END;

                     WHILE C <> NIL DO
                       BEGIN
                         NEW(PTR);
                         WRITELN('RECEIVER ID ?');
                         READLN( A );
                           IF ( A = 0 ) THEN
                             BEGIN
                               CURRENT^.NEXT := NIL;
                               DISPOSE(PTR);
                               C := NIL;
                             END
                            ELSE
                             BEGIN
                               WRITELN;
                               WRITELN('PROBABILITY ?');
                               READLN( B );
                               WRITELN;
                               PTR^.ID := A;
                               PTR^.PROB := B;
                               CURRENT^.NEXT := PTR;
                               CURRENT := PTR;
                               C := PTR;
                             END;
                            END;

                       END; [ FOR K ]
                   END; [ IF NOT(OPEN) ]
                 END; [ WITH ADAPTER() ]
               END; [ ?WFOR J=1 TO 4 ]
```

```
769                 END; [ FOR I=1 TO NUM OF ADAPTERS ]
770               END;
771             TEMP := 0.0;
772             TEMP1 := 0.0;
773             A := 0;
774             FOR I := 1 TO COMMAD DO
775                 BEGIN
776                   A := ADAPTER[I].P1;
777                  SORT[A] := I;
778                 END;
779
780             FOR I := 1 TO COMMAD DO
781               WITH ADAPTER[I] DO
782                 BEGIN
783                   IF I =1 THEN
784                     ADAPTER[SORT[I]].PRI1DELAY := 0.48E-06
785                   ELSE
786                     ADAPTER[SORT[I]].PRI1DELAY := ADAPTER[SORT[I-1]].PRI1DELAY +
787                               ABS(ADAPTER[SORT[I]].DISTFROMA1 -
788                               ADAPTER[SORT[I-1]].DISTFROMA1) * 4.0E-09 + 1.6E-06;
789                 END;
790
791           FOR I := 1 TO COMMAD DO
792             BEGIN
793               TEMP := ADAPTER[I].DISTFROMA1;
794                 FOR J := 1 TO COMMAD DO
795                   IF TEMP <= ADAPTER[J].DISTFROMA1 THEN
796                       TEMP := ADAPTER[J].DISTFROMA1;
797
798           WITH ADAPTER[I] DO
799             BEGIN
800               ADAPTER[I].END1DELAY := ADAPTER[SORT[COMMAD]].PRI1DELAY +
801                                       TEMP * 4.0E-09 + 1.6E-06;
802                           TEMP1 := ADAPTER[I].END1DELAY;
803                     IF TEMP1 <= ADAPTER[I].END1DELAY THEN
804                        TEMP1 := ADAPTER[I].END1DELAY;
805             END;
806           END;
807
808           FOR I := 1 TO COMMAD DO
809               WITH ADAPTER[I] DO
810                 BEGIN
811                   ADAPTER[I].END1DELAY := TEMP1;
812                   ADAPTER[I].TOT1DELAY := FIX1DELAY + TEMP1;
813                 END;
814
815
816              IF TRUNKS THEN
817                 BEGIN
818                     FOR I := (COMMAD + 1 ) TO NUMOFADAPTERS DO
819                      WITH ADAPTER[I] DO
820                      BEGIN
821                        WRITELN;
822                        WRITELN('ADAPTER #', I: 2, ':');
823                        WRITELN;
```

```
824                 WRITELN(' ' :2,
825                    'DISTANCE IN FT FROM PRIORITY 1 ADAPTER OF TRUNK 2 : ');
826                 READLN( DISTFRTA1);
827
828                 WRITELN;
829                 WRITELN(' ':2,
830                     'ADAPTER PRIORITY : GIVE IN INTEGER FORM,IE. N= 1,2,..');
831                 WRITELN(' ':2, 'THIS IS USED TO CALCULATE PRIORITY DELAY');
832                 READLN( P2);
833                 FIX2DELAY := TRUNKLENGTH[2] * 4.0E-09 + 2.0EE-06;
834                 WRITELN;
835                 WRITELN;
836                 WRITELN(' ':2,
837                     ' ADAPTER RETRY COUNT : THIS MUST BE A UNIQUE NUMBER');
838                 WRITELN(' ':25,'FOR EACH ADAPTER IN THE RANGE OF 0 TO 64');
839                 READLN ( RETRYCT );
840                 WRITELN;
841                 FOR J := 1 TO 4 DO
842                  BEGIN
843                   WITH ADAPTER[I].DEVICE[J] DO
844                    BEGIN
845                     OPEN := TRUE;
846                     WRITELN;
847                     WRITELN(' ' :2,'ADAPTER ',I:2,' DEVICE ',J:2,
848                         ' DESCRIPTION ');
849                     WRITELN(' ' :4,'DEVICE STATUS: O)PEN OR C)LOSED ');
850                     REPEAT READ (RESP[1]) UNTIL RESP[1] <> ' ';READLN;
851                     IF (RESP[1] = 'C') OR (RESP[1] = 'C') THEN
852                      OPEN := FALSE;
853                     IF NOT(OPEN) THEN
854                       BEGIN
855                        WRITELN(' ':4,'DEVICE ID (<=24 CHAR)');
856                        READLN( ID );
857
858                        WRITELN(' ':4,
859                            'DEVICE TO ADAPTER I/O RATE (BYTES/SEC):');
860                        READLN (TFERRATE);
861                        TFERRATE := TFERRATE * 8;
862                        LOADTIME := 1.0/TFERRATE;
863
864                        WRITELN(' ':4,
865                            'NUMBER OF DATA SOURCES IN DEVICE ',J:1);
866                        READLN( NUMOFSOURCES );
867
868                        FOR K := 1 TO NUMOFSOURCES DO
869                         WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
870                          BEGIN
871                           DISTFRTA1 := ADAPTER[I].DISTFRTA1;
872                           WRITELN;
873                           WRITELN(' ':5,'DATA SOURCE', K:1, ' ID');
874                           READLN( ID );
875                           DNUM := I * 100 + J * 10 + K;
876                           WRITELN;
877                           WRITELN(' ':5,'SOURCE[',K:1,
878                               '] DATA GEN RATE (BYTES/SEC)');
```

```
879                     READLN( GENRATE );
880                     GENRATE := GENRATE * 8;
881                     WRITELN(' ':5, 'BUFFER SIZE IN BYTES:');
882                     READLN( BUFFERSIZE );
883                     BUFFERSIZE := BUFFERSIZE * 8;
884                     DATABLKCT := BUFFERSIZE / 16384;
885                     TFERRATE := ADAPTER[I].DEVICE[J].TFERRATE;
886                     OFFLOAD := OFFLOAD + GENRATE;
887                     IF GENRATE > 0.0 THEN
888                        TXINTRVL := BUFFERSIZE / GENRATE
889                     ELSE
890                        TXINTRVL := 1E7;
891
892                        NEXTTX := TXINTRVL;
893
894                  WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
895                    BEGIN
896                     WRITELN(' ':2,
897                         'AS BEFORE A LIST OF RECEIVERS, PROBABILITY ');
898                     WRITELN(' ':2,
899                         'WILL BE MADE. USE  0  FOR ID TO FINISH LIST');
900                     A := 0;
901                     B := 0.0;
902                     CURRENT := FIRST;
903                     WRITELN('RECEIVER ID CODE ? ( 0 TO END LIST)');
904                     READLN( A );
905                     WRITELN;
906                      IF A = 0 THEN
907                       BEGIN
908                         CURRENT := NIL;
909                         C := CURRENT;
910                         FIRST := CURRENT;
911                       END
912                      ELSE
913                       BEGIN
914                         WRITELN('PROBABILITY ?');
915                         READLN( B );
916                       CURRENT^.ID := A;
917                       CURRENT^.PROB := B;
918                         C := CURRENT;
919                       END;
920                    WHILE C <> NIL DO
921                      BEGIN
922                        NEW(PTR);
923                        WRITELN('RECEIVER ID ? ( 0 TO END LIST)');
924                        READLN( A );
925                         IF A = 0 THEN
926                           BEGIN
927                             CURRENT^.NEXT := NIL;
928                             C := NIL; DISPOSE(PTR);
929                           END
930                         ELSE
931                           BEGIN
932                             WRITELN;
933                             WRITELN('PROBABILITY ?');
```

```
                                    READLN( B );
                                    PTR^.ID := A;
                                    PTR^.PROB := B;
                                    CURRENT^.NEXT := PTR;
                                    CURRENT := PTR;
                                    C := PTR;
                                  END;
                              END;
                            END;
                          END;
                        END;
                      END;
                    END;
                  END;
                END;

          IF TRUNKS THEN
            BEGIN
           A := 0;
           FOR I := COMMAD TO NUMOFADAPTERS DO
              BEGIN
                  A := ADAPTER[I].P2 + ( COMMAD - 1 );
                  SORT1[A] := I;
              END;

           FOR I := COMMAD TO NUMOFADAPTERS DO
            WITH ADAPTER[I] DO
             BEGIN
              IF ( I = COMMAD ) THEN
                 ADAPTER[SORT1[I]].PRI2DELAY := 0.48E-06
              ELSE
                 ADAPTER[SORT1[I]].PRI2DELAY := ADAPTER[SORT1[I-1]].PRI2DELAY +
                          ABS(ADAPTER[SORT1[I]].DISTFRTA1 -
                              ADAPTER[SORT1[I - 1]].DISTFRTA1) * 4.0E-09 + 1.6E-06;
             END;

           TEMP := 0.0;
           TEMP1 := 0.0;

           FOR I := COMMAD TO NUMOFADAPTERS DO
              BEGIN
                TEMP := ADAPTER[I].DISTFRTA1;
                  FOR J := COMMAD TO NUMOFADAPTERS DO
                       IF TEMP <= ADAPTER[J].DISTFRTA1 THEN
                         TEMP := ADAPTER[J].DISTFRTA1;
                WITH ADAPTER[I] DO
                  BEGIN
                    ADAPTER[I].END2DELAY := ADAPTER[SORT1[NUMOFADAPTERS]].PRI2DELAY
                                            + 4.0E-09 * TEMP + 1.6E-06;
                    TEMP1 := ADAPTER[I].END2DELAY;
                     IF TEMP1 <= ADAPTER[I].END2DELAY THEN
                        TEMP1 := ADAPTER[I].END2DELAY;
                  END;
              END;
```

```
989             FOR I := COMMAD TO NUMOFADAPTERS DO
990                 BEGIN
991                   ADAPTER[I].END2DELAY := TEMP1;
992                   ADAPTER[I].TOT2DELAY := TEMP1 + ADAPTER[I].FIX2DELAY;
993                 END;
994              END;
995
996
997
998               REWRITEDFILE;
999               END; [ INTERACTIVE CASE ]
1000
1001            FILEINPUT:
1002              BEGIN
1003              (*OPEN(DFILE,'DESCRIP',OLD);*)
1004              RESET(DFILE);
1005
1006              READLN(DFILE, NUMOFTRUNKS);
1007              FOR I := 1 TO NUMOFTRUNKS DO READLN(DFILE, TRUNKLENGTH[I]);
1008
1009              READLN(DFILE, NUMOFADAPTERS);
1010              READLN(DFILE, COMMAD );
1011              FOR I := 1 TO COMMAD DO
1012                BEGIN
1013                WITH ADAPTER[I] DO
1014                  BEGIN
1015                  READLN(DFILE, DISTFROMA1);
1016                  READLN(DFILE, PRI1DELAY    );
1017                  READLN(DFILE, END1DELAY    );
1018                  FIX1DELAY  := TRUNKLENGTH[1] * 4.0E-09 + 2.08E-06;
1019                  TOT1DELAY  := FIX1DELAY + END1DELAY ;
1020                  READLN(DFILE, RETRYCT );
1021                   IF ( TRUNKS ) AND ( I = COMMAD ) THEN
1022                    BEGIN
1023                      READLN(DFILE, DISTFRTA1);
1024                      READLN(DFILE, PRI2DELAY);
1025                      READLN(DFILE, END2DELAY);
1026                      FIX2DELAY := TRUNKLENGTH[2] * 4.0E-09 + 2.08E-06;
1027                      TOT2DELAY := FIX2DELAY + END2DELAY;
1028                    END;
1029
1030                  [   BEGIN INDIVIDUAL DEVICE DESCRIPTIONS   ]
1031
1032                  FOR J := 1 TO 4 DO
1033                    BEGIN
1034                    WITH ADAPTER[I],DEVICE[J] DO
1035                      BEGIN
1036                      REPEAT
1037                        READ(DFILE, RESP[1])
1038                      UNTIL RESP[1] <> ' '; READLN(DFILE);
1039                      IF (RESP[1] = 'F') OR (RESP[1] = 'F') THEN
1040                        OPEN := FALSE
1041                      ELSE
1042                        OPEN := TRUE;
1043                      IF NOT OPEN THEN
```

```
1044                    BEGIN
1045
1046                    READLN(DFILE, ID);
1047
1048                    READLN(DFILE, TFERRATE);
1049                    IF TFERRATE <> 0.0 THEN
1050                      LOADTIME := 1.0 / TFERRATE
1051                    ELSE
1052                      LOADTIME := 0.0;
1053
1054                    READLN(DFILE, NUMOFSOURCES);
1055
1056                    FOR K := 1 TO NUMOFSOURCES DO
1057                      WITH ADAPTER[I],DEVICE[J],SOURCE[K] DO
1058                        BEGIN
1059                        OPEN := FALSE;
1060                        DISTFROMA1 := ADAPTER[I].DISTFROMA1;
1061                        IF (I = COMMAD ) AND (TRUNKS) THEN
1062                        DISTFRTA1  := ADAPTER[I].DISTFRTA1;
1063                        READLN(DFILE, DNUM  );
1064                        READLN(DFILE, ID);
1065                        READLN(DFILE, GENRATE);
1066                        READLN(DFILE, BUFFERSIZE);
1067                        READLN(DFILE, DATABLKCT );
1068                        TFERRATE := ADAPTER[I].DEVICE[J].TFERRATE;
1069                        OFFLOAD := OFFLOAD + GENRATE;
1070                        IF GENRATE > 0.0 THEN
1071                          TXINTRVL := BUFFERSIZE / GENRATE
1072                        ELSE
1073                          TXINTRVL := 1E7;
1074
1075                        NEXTTX := TXINTRVL;
1076                       WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1077                         BEGIN
1078                           CURRENT := FIRST;
1079                           READLN(DFILE, A );
1080                            IF A = 0 THEN
1081                             BEGIN
1082                              CURRENT := NIL;
1083                              FIRST := CURRENT;
1084                             END
1085                            ELSE
1086                              BEGIN
1087                                READLN(DFILE, B );
1088                              CURRENT^.ID := A;
1089                              CURRENT^.PROB := B;
1090                               WHILE A <> 0 DO
1091                                 BEGIN
1092                                   NEW(PTR);
1093                                   READLN(DFILE, A );
1094                                 IF A = 0 THEN
1095                                   BEGIN
1096                                     CURRENT^.NEXT := NIL;
1097                                     DISPOSE (PTR);
1098                                   END
```

```
1099                                ELSE
1100                                  BEGIN
1101                                    READLN(DFILE, B );
1102                                    PTR^.ID := A;
1103                                    PTR^.PROB := B;
1104                                    CURRENT^.NEXT := PTR;
1105                                    CURRENT := PTR;
1106                                  END;
1107                              END;
1108                            END;
1109                          END;
1110                      END; [ FOR K:=1 TO NUM OF SOURCES ]
1111                  END; [ IF NOT(OPEN) ]
1112              END; [ WITH ADAPTER().DEVICE() ]
1113          END; [ FOR J=1 TO 4 ]
1114      END; [ ?WWITH ADAPTER() ]
1115  END; [ FOR I=1 TO NUM OF ADAPTERS ]
1116
1117  IF TRUNKS THEN
1118    BEGIN
1119     FOR I := (COMMAD + 1 ) TO NUMOFADAPTERS DO
1120      BEGIN
1121       WITH ADAPTER[I] DO
1122        BEGIN
1123         READLN(DFILE, DISTFRTA1 );
1124         READLN(DFILE, PRI2DELAY );
1125         READLN(DFILE, END2DELAY );
1126         FIX2DELAY := TRUNKLENGTH[2] * 4.0E-09 + 2.08E-06;
1127         TOT2DELAY := FIX2DELAY + END2DELAY;
1128         READLN(DFILE, RETRYCT );
1129
1130         (*  BEGIN INDIVIDUAL DEVICE DESCRIPTIONS *)
1131
1132         FOR J := 1 TO 4 DO
1133          BEGIN
1134           WITH ADAPTER[I].DEVICE[J] DO
1135            BEGIN
1136             REPEAT
1137              READ (DFILE,RESP[1]) UNTIL
1138               RESP[1] <> ' ';READLN(DFILE);
1139             IF (RESP[1] = 'F') OR (RESP[1] = 'f') THEN
1140               OPEN := FALSE
1141             ELSE
1142               OPEN := TRUE;
1143             IF NOT(OPEN) THEN
1144               BEGIN
1145                READLN(DFILE,ID);
1146                READLN(DFILE, TERRATE );
1147                IF TERRATE <> 0.0 THEN
1148                  LOADTIME := 1.0 / TERRATE
1149                ELSE
1150                  LOADTIME := 0.0;
1151                READLN(DFILE, NUMOFSOURCES );
1152                FOR K := 1 TO NUMOFSOURCES DO
1153                 WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
```

```
1154                    BEGIN
1155                      DISTFRTA1 := ADAPTER[I].DISTFRTA1;
1156                      READLN(DFILE, DNUM);
1157                      READLN(DFILE, ID );
1158                      READLN(DFILE, GENRATE );
1159                      READLN(DFILE, BUFFERSIZE );
1160                      TFERRATE :=  ADAPTER[I].DEVICE[J].TFERRATE;
1161                      READLN(DFILE, DATABLKCT );
1162                      OFFLOAD := OFFLOAD + GENRATE;
1163                      IF GENRATE > 0.0 THEN
1164                        TXINTRVL := BUFFERSIZE / GENRATE
1165                      ELSE
1166                        TXINTRVL := 1.0E7;
1167
1168                      NEXTTX := TXINTRVL;
1169
1170                     WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1171                       BEGIN
1172                         CURRENT := FIRST;
1173                         READLN(DFILE, A );
1174                          IF A = 0 THEN
1175                           BEGIN
1176                            CURRENT := NIL;
1177                            FIRST := CURRENT;
1178                           END
1179                          ELSE
1180                            BEGIN
1181                              READLN(DFILE, B );
1182                            CURRENT^.ID := A;
1183                            CURRENT^.PROB := B;
1184                            WHILE A <> 0 DO
1185                              BEGIN
1186                                NEW( PTR );
1187                                READLN(DFILE, A );
1188                               IF A = 0 THEN
1189                                 BEGIN
1190                                   CURRENT^.NEXT := NIL;
1191                                   DISPOSE(PTR);
1192                                 END
1193                               ELSE
1194                                 BEGIN
1195                                   READLN(DFILE, B );
1196                                   PTR^.ID := A;
1197                                   PTR^.PROB := B;
1198                                   CURRENT^.NEXT := PTR;
1199                                   CURRENT := PTR;
1200                                 END;
1201                                END;
1202                              END;
1203                            END;
1204
1205                       END;
1206                    END;
1207                  END;
1208                END;
```

```
1209                END;
1210              END;
1211            END;
1212          END; [ FILE_INPUT CASE ]
1213
1214        END; [ CASE ]
1215
1216    END; [ PROCEDURE CHARACTERIZE NETWORK ]
1217
1218
1219
1220
1221  [*************************************************]
1222  [                                                 ]
1223
1224
1225  PROCEDURE PRINTNETDESCRIPTION;
1226   [          PRINTS THE DESCRIPTION OF THE ENTIRE       ]
1227   [          NETWORK IN AUX-OUTPUT FILE.                ]
1228   [*************************************************]
1229
1230    VAR
1231      I, L, J, K, KK, LL, S  : INTEGER;
1232
1233    BEGIN [ PROCEDURE PRINT DESCRIPTION ]
1234
1235      (*OPEN(AUXOUT,'AUXOUT',OLD);*)
1236      REWRITE(AUXOUT);
1237      FOR I := 1 TO COMMAD DO
1238        BEGIN
1239
1240        WRITELN(AUXOUT);
1241        WRITELN(AUXOUT, ' ': 10,
1242                '*************************************************');
1243        WRITELN(AUXOUT, ' ': 10,
1244                '          ***************************************');
1245        WRITELN(AUXOUT);
1246        WRITELN(AUXOUT, ' ': 10, '                    NETWORK DESCRIPTION');
1247        WRITELN(AUXOUT, ' ': 31, 'ADAPTER #', I: 3);
1248        WRITELN(AUXOUT);
1249        WRITELN(AUXOUT, ' ': 10,
1250                '          ***************************************');
1251        WRITELN(AUXOUT, ' ': 10,
1252                '*************************************************');
1253
1254        WITH ADAPTER[I] DO
1255          BEGIN
1256          WRITELN(AUXOUT);
1257          WRITELN(AUXOUT);
1258          WRITELN(AUXOUT, 'ADAPTER #', I: 2, ':');
1259          J := 1;
1260
1261          WRITELN(AUXOUT);
1262
1263          WRITE(AUXOUT, ' ': 4, 'PRIORITY DELAY ON TRUNK 1 : ');
```

```
1264            WRITELN(AUXOUT, PRI1DELAY    : 9: 7, ' SEC');
1265            WRITE(AUXOUT, ' ': 4, 'FIXED DELAY: ');
1266            WRITELN(AUXOUT, FIX1DELAY : 9: 7, 'SEC');
1267            WRITE  (AUXOUT,' ':4, 'TOTALDELAY ON TRUNK 1 : ');
1268            WRITELN(AUXOUT,TOT1DELAY: 9: 7, 'SECS');
1269
1270            WRITE  (AUXOUT,' ':4, 'ADAPTER RETRY COUNT :');
1271            WRITELN(AUXOUT, RETRYCT :3);
1272
1273
1274                 IF ( TRUNKS ) AND ( I = COMMAD ) THEN
1275                    BEGIN
1276                      WRITELN(AUXOUT);
1277                      WRITE  (AUXOUT,' ':4,'PRIORITY DELAY ON TRUNK 2 : ');
1278                      WRITELN(AUXOUT, PRI2DELAY : 9: 7, 'SEC');
1279                      WRITE  (AUXOUT,' ':4, 'FIXED DELAY: ');
1280                      WRITELN(AUXOUT, FIX2DELAY : 9: 7, 'SEC');
1281                      WRITE  (AUXOUT,' ':4,'TOTAL DELAY ON TRUNK 2 : ');
1282                      WRITELN(AUXOUT, TOT2DELAY : 9: 7, 'SEC');
1283                    END;
1284            {   BEGIN INDIVIDUAL DEVICE DESCRIPTIONS   }
1285
1286            FOR J := 1 TO 4 DO
1287              BEGIN
1288              WITH ADAPTER[I],DEVICE[J] DO
1289                BEGIN
1290                WRITELN(AUXOUT);
1291                WRITE(AUXOUT, ' ': 4, 'DEVICE ', J: 2);
1292                WRITE(AUXOUT, ' ': 2, ' STATUS: ');
1293                IF OPEN THEN
1294                  BEGIN
1295                  WRITELN(AUXOUT, 'OPEN');
1296                  END
1297                ELSE
1298                  WRITE(AUXOUT, 'CLOSED');
1299                IF NOT (OPEN) THEN
1300                  BEGIN
1301                  WRITELN(AUXOUT, '--': 2, ID);
1302
1303                  WRITE(AUXOUT, ' ': 32, 'I/O BUS TRANSFER RATE:');
1304                  WRITELN(AUXOUT, XFERRATE / 8: 12: 2, ' BYTES/SEC ');
1305
1306                  WRITELN(AUXOUT, ' ': 32, 'LOAD TIME: ', LOADTIME: 12: 7,
1307                          ' SEC');
1308
1309                  WRITE(AUXOUT, ' ': 32, 'NUMBER OF DATA SOURCES:');
1310                  WRITELN(AUXOUT, NUMOFSOURCES: 3);
1311
1312                  FOR K := 1 TO NUMOFSOURCES DO
1313                    WITH ADAPTER[I],DEVICE[J],SOURCE[K] DO
1314                      BEGIN
1315                      WRITELN(AUXOUT, ' ': 32, ID);
1316                      WRITELN(AUXOUT, ' ': 34, 'DEVICE NUMBER ', DNUM  : 4);
1317                      WRITE(AUXOUT, ' ': 34, 'BUFFER SIZE: ');
1318                      WRITELN(AUXOUT, BUFFERSIZE /   8: 8, ' BYTES');
```

```
1319
1320                    WRITE(AUXOUT, ' ': 34, 'SOURCE #', K: 2,
1321                          ' DATA GENERATION RATE: ');
1322                    WRITELN(AUXOUT, SOURCE[K].GENRATE / 8: 9: 2, ' BYTES');
1323                    WRITELN(AUXOUT, ' ': 34, 'TRUNK TRANSMISSION INTERVAL: ',
1324                            TXINTRVL: 7: 4, ' SEC');
1325
1326                    WRITELN(AUXOUT,' ':34, 'DATA BLOCK COUNT : ',DATABLKCT);
1327
1328                    WRITE  (AUXOUT,'POSSIBLE RECEIVERS AND THEIR',
1329                                   ' RESPECTIVE PROBABILITIES FOR:' ,DNUM:3);
1330                    WRITELN(AUXOUT);
1331                    WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1332                      BEGIN
1333                       C := FIRST;
1334                        WHILE C <> NIL DO
1335                         BEGIN
1336                           WRITE  (AUXOUT,'RECEIVER ID: ',C^.ID:3);
1337                           WRITELN(AUXOUT,' ':20,'PROBABILITY: ',C^.PROB:5  );
1338                           C := C^.NEXT;
1339                         END;
1340                      END;
1341                    END; { FOR K:=1 TO NUM OF SOURCES }
1342                 END; { IF NOT(OPEN) }
1343               END; { WITH ADAPTER().DEVICE() }
1344             END; { FOR J=1 TO 4 }
1345           END; { ?WWITH ADAPTER() }
1346           FOR S := 1 TO 5 DO WRITELN(AUXOUT);
1347         END; { FOR I }
1348
1349       IF TRUNKS THEN
1350         BEGIN
1351           FOR I := ( COMMAD + 1 ) TO NUMCFADAPTERS DO
1352            BEGIN
1353            WRITELN(AUXOUT);
1354            WRITELN(AUXOUT,' ':10,
1355                    '*****************************************************');
1356            WRITELN(AUXOUT,' ':10,
1357                    '  ***************************************');
1358            WRITELN(AUXOUT);
1359            WRITELN(AUXOUT,' ':10,'                    NETWORK DESCRIPTION');
1360            WRITELN(AUXOUT,' ':31,'ADAPTER # ', I:3  );
1361            WRITELN(AUXOUT);
1362            WRITELN(AUXOUT,' ':10,
1363                    '  ***************************************');
1364            WRITELN(AUXOUT,' ':10,
1365                    '*****************************************  ***************');
1366
1367            WITH ADAPTER[I] DO
1368             BEGIN
1369              WRITELN(AUXOUT);
1370              WRITELN(AUXOUT);
1371              WRITELN(AUXOUT, 'ADAPTER # ' ,I:2, ':');
1372              J := 1;
1373
```

```
1374              WRITELN(AUXOUT);
1375              WRITE  (AUXOUT,' ':4, 'PRIORITY DELAY ON TRUNK 2 : ');
1376              WRITELN(AUXOUT, PRI2DELAY: 9: 7, 'SECS');
1377              WRITE  (AUXOUT,' ':4, 'FIXED DELAY ON TRUNK 2: ');
1378              WRITELN(AUXOUT,FIX2DELAY: 9: 7, 'SECS');
1379              WRITE  (AUXOUT,' ':4, 'TOTAL DELAY ON TRUNK 2: ');
1380              WRITELN(AUXOUT,TOT2DELAY: 9: 7, 'SECS');
1381              WRITE  (AUXOUT,' ':4, 'ADAPTER RETRY COUNT : ');
1382              WRITELN(AUXOUT, RETRYCT : 3);
1383
1384
1385              FOR J := 1 TO 4 DO
1386                 BEGIN
1387                   WITH ADAPTER[I].DEVICE[J] DO
1388                     BEGIN
1389                       WRITELN(AUXOUT);
1390                       WRITE  (AUXOUT,' ':4, 'DEVICE ',J:2);
1391                       WRITE  (AUXOUT,' ':2, 'STATUS : ');
1392                       IF OPEN THEN
1393                         BEGIN
1394                          WRITELN(AUXOUT, 'OPEN');
1395                         END
1396                       ELSE
1397                          WRITE  (AUXOUT, 'CLOSED');
1398                      IF NOT(OPEN) THEN
1399                        BEGIN
1400                        WRITELN(AUXOUT, '--':2, ID );
1401
1402                        WRITE  (AUXOUT,' ':32, 'I/O BUS TRANSFER RATE:');
1403                        WRITELN(AUXOUT, TFERRATE / 8:12:2, ' BYTES/SEC ');
1404
1405                        WRITELN(AUXOUT,' ':32, 'LOAD TIME: ', LOADTIME:12:7,
1406                                ' SEC');
1407
1408                        WRITE  (AUXOUT,' ':32, 'NUMBER OF DATA SOURCES:');
1409                        WRITELN(AUXOUT, NUMOFSOURCES: 3);
1410
1411                        FOR K := 1 TO NUMOFSOURCES DO
1412                          WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1413                           BEGIN
1414                            WRITELN(AUXOUT,' ':32, ID);
1415                            WRITELN(AUXOUT,' ':34, 'DEVICE NUMBER ',DNUM :4);
1416                            WRITE  (AUXOUT,' ':34, 'BUFFER SIZE: ');
1417                            WRITELN(AUXOUT, BUFFERSIZE / 8:8, 'BYTES');
1418
1419                            WRITE  (AUXOUT,' ':34, 'SOURCE #', K:2,
1420                                                   'DATA GENERATION RATE: ');
1421                            WRITELN(AUXOUT, SOURCE[K].GENRATE / 8: 9: 2, 'BYTES');
1422                            WRITELN(AUXOUT,' ':34,'TRUNK TRANSMISSION INTERVAL: ',
1423                                      TXINTRVL: 7: 4, 'SECS');
1424                           WRITELN(AUXOUT,'POSSIBLE RECEIVERS AND THEIR ',
1425                                   'RESPECTIVE PROBABILITIES FOR: ',DNUM:3);
1426                          WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1427                            BEGIN
1428                             C := FIRST;
```

```
1429                              WHILE C <> NIL DO
1430                               BEGIN
1431                                  WRITE  (AUXOUT,'RECEIVER ID: ',C^.ID:3);
1432                                  WRITELN(AUXOUT,' ':20,'PROBABILITY: ',
1433                                          C^.PROB: 5   );
1434                                  C := C^.NEXT;
1435                               END;
1436                             END;
1437                           END;
1438                         END;
1439                       END;
1440                     END;
1441                   END;
1442                   FOR S := 1 TO 5 DO WRITELN(AUXOUT);
1443                 END;
1444       END; { PROCEDURE PRINT DESCRIPTION }
1445
1446   (*************************************************)
1447   (*                                               *)
1448
1449       FUNCTION UNIFORM(VAR SEED: INTEGER): REAL;
1450   (*                                               *)
1451   (*     CALCULATES A PSEUDORANDOM NUMBER IN THE   *)
1452   (*     RANGE OF 0 < NUM < 1.                     *)
1453   (*                                               *)
1454   (*************************************************)
1455
1456      CONST
1457         L = 29;
1458         C = 217;
1459         M = 1024;
1460       BEGIN
1461          SEED := ((SEED * L) + C) MOD M;
1462          UNIFORM := SEED / M;
1463       END;
1464
1465
1466
1467     {*************************************************}
1468     {                                                 }
1469
1470
1471     PROCEDURE FINDNEXT(VAR TMITTER: SOURCEDESCRIPTIONS);
1472      {    DETERMINES THE NEXT TRANSMITTER BY EXAMINING     }
1473      {    EACH DEVICE'S TIME UNTIL NEXT TRANSMISSION.      }
1474      {    NEXT TMITTR IS DEVICE WITH SHORTEST TIME UNTIL   }
1475      {    NEXT TRANSMISSION.                               }
1476      {                                                     }
1477      {*****************************************************}
1478
1479       VAR
1480         I, J, K,L, TA, TD, TS  : INTEGER;
1481         TMIT  : DEVICERECORDS;
1482         PROB                   : REAL;
1483
```

```
1484
1485        BEGIN
1486
1487          RESERVED := FALSE;
1488          PROB := UNIFORM(U);
1489
1490          TMITTER.NEXTTX := 1000.0;
1491          FOR I := 1 TO NUMOFADAPTERS DO
1492            FOR J := 1 TO 4 DO
1493              WITH ADAPTER[I],DEVICE[J] DO
1494                IF NOT OPEN THEN
1495                  FOR K := 1 TO NUMOFSOURCES DO
1496                    WITH ADAPTER[I],DEVICE[J],SOURCE[K] DO
1497                      IF NEXTTX < TMITTER.NEXTTX THEN
1498                        TMITTER := ADAPTER[I].DEVICE[J].SOURCE[K];
1499
1500          TA := TMITTER.DNUM DIV 100;
1501          TD := (TMITTER.DNUM MOD 100) DIV 10;
1502          TS := TMITTER.DNUM MOD 10;
1503
1504          WITH ADAPTER[TA] DO
1505          WITH TMITTER DO
1506            BEGIN
1507              FOR L := 1 TO 4 DO
1508               IF((DFLAG[L]) AND (L <> TD)) OR (FROMTX <> 0) THEN
1509                  RESERVED := TRUE;
1510             END;
1511          WITH ADAPTER[TA] DO
1512           WITH ADAPTER[TA].DEVICE[TD] DO
1513
1514          WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1515          IF RESERVED THEN
1516           BEGIN
1517              NEXTTX := NEXTTX + (PROB * 10.0E-05);
1518              WAITTIME :=  WAITTIME + (PROB * 10.0E-05);
1519              WAITCT := WAITCT + 1;
1520              WAITTALLY := WAITTALLY + 1;
1521         END;
1522          END; (* FIND NEXT TRANSMITTER *)
1523
1524
1525
1526
1527
1528       (*****************************************************)
1529       [                                                   ]
1530
1531
1532      PROCEDURE PICKA(VAR TMITTER : SOURCEDESCRIPTIONS);
1533       [     PICKS AN ELIGIBLE RECEIVER FOR THE CURRENT      ]
1534       [     TRANSMITTER, BASED ON THE LIST OF POSSIBLE      ]
1535       [     RECEIVERS GIVEN IN NETWORK DESCRIPTION FOR      ]
1536       [     EACH ADAPTER/DEVICE AND SOURCE.                 ]
1537       [*****************************************************]
1538
```

```
1539        VAR
1540          I, J, K                  : INTEGER;
1541          FOUND: BOOLEAN;
1542          PROB: REAL;
1543
1544        BEGIN [ PROCEDURE TO PICK A RECEIVER ]
1545
1546          PROB := 0.0;
1547          FOUND := FALSE;
1548
1549          WHILE NOT FOUND DO
1550            BEGIN
1551             PROB := UNIFORM (U);
1552              WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1553               BEGIN
1554                 C := FIRST;
1555                  WHILE C <> NIL DO
1556                   IF C^.PROB > PROB THEN
1557                     BEGIN
1558                      FOUND := TRUE ;
1559                      DEST := C^.ID;
1560                      C := NIL;
1561                     END
1562                   ELSE
1563                     C := C^.NEXT;
1564               END;
1565            END;
1566
1567           WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1568            BEGIN
1569             IF TA < COMMAD THEN
1570              IF (DEST DIV 100) > COMMAD THEN
1571                IDEST := COMMAD
1572              ELSE
1573                IDEST := 0;
1574
1575              IF TA > COMMAD THEN
1576               IF (DEST DIV 100) < COMMAD THEN
1577                IDEST := COMMAD
1578               ELSE
1579                IDEST := 0;
1580          END;
1581        END; (* PICKA *)
1582
1583    (*************************************************************************)
1584
1585     FUNCTION ACOLLISION : BOOLEAN;
1586
1587    (*                      DETERMINES WHETHER OR NOT A COLLISION HAS        *)
1588    (*                      OCCURED ON THE TRUNK UNDER CONSIDERATION.        *)
1589    (*************************************************************************)
1590      VAR
1591        I, J, K      : INTEGER;
1592        TEMP               : REAL;
1593
```

```
1594        BEGIN
1595          TEMP := 0.0;
1596          TIMEDIF := 0.0;
1597          ACOLLISION := FALSE;
1598
1599          IF TRK1 THEN
1600           BEGIN
1601            FOR I := 1 TO COMMAD DO
1602             FOR J := 1 TO 4 DO
1603              WITH ADAPTER[I].DEVICE[J] DO
1604               IF NOT (OPEN) THEN
1605                FOR K := 1 TO NUMOFSOURCES DO
1606                 WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1607                  BEGIN
1608                  IF TMITTER.DNUM DIV 100 <> I THEN
1609                   IF (M1) OR (M2) OR (M3) OR (M4) OR (M5) OR (M6) THEN
1610                    BEGIN
1611                     TEMP := ABS( TMITTER.DISTFROMA1 - DISTFROMA1 );
1612                     TIMEDIF := 2 * (TEMP * PROPAGATION);
1613                     IF ABS( TMITTER.NEXTTX - NEXTTX ) <= TIMEDIF THEN
1614                     WITH ADAPTER[I] DO
1615                      WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1616                      BEGIN
1617                        ACOLLISION := TRUE;
1618                        COLLTALLY := COLLTALLY + 1;
1619                        COLLTIME := COLLTIME + TIMEDIF;
1620                        COLLCT := COLLCT + 1;
1621                        NEXTTX := NEXTTX + FIX1DELAY + PRI1DELAY;
1622                      END;
1623                    END;
1624                  END;
1625           END
1626          ELSE
1627           BEGIN
1628            FOR I := COMMAD TO NUMOFADAPTERS DO
1629             FOR J := 1 TO 4 DO
1630              WITH ADAPTER[I].DEVICE[J] DO
1631               IF NOT (OPEN) THEN
1632                FOR K := 1 TO NUMOFSOURCES DO
1633                 WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1634                  BEGIN
1635                  IF TMITTER.DNUM DIV 100 <> I THEN
1636                   IF (M1) OR (M2) OR (M3) OR (M4) OR (M5) OR (M6) THEN
1637                    BEGIN
1638                     TEMP := ABS( TMITTER.DISTFRTA1 - DISTFRTA1 );
1639                     TIMEDIF := 2 * ( TEMP * PROPAGATION);
1640                     IF ABS( TMITTER.NEXTTX - NEXTTX ) <= TIMEDIF THEN
1641                     WITH ADAPTER[I] DO
1642                      WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
1643                      BEGIN
1644                       ACOLLISION := TRUE;
1645                       COLLTALLY := COLLTALLY + 1 ;
1646                       COLLTIME := COLLTIME + TIMEDIF ;
1647                       COLLCT := COLLCT + 1;
1648                       NEXTTX := NEXTTX + FIX2DELAY + PRI2DELAY;
```

```
1649                    END;
1650                  END;
1651                END;
1652          END;
1653     END; (* ACOLLISION *)
1654    (*******************************************************)
1655
1656          PROCEDURE UPDATE(COND : CLOCKCOND );
1657    (*          THERE ARE SEVEN CASES CONTAINED IN THIS        *)
1658    (*          PROCEDURE. EACH ONE IS RESPONSIBLE FOR UP-     *)
1659    (*          DATING AN ADAPTER'S VARIOUS STATISTICS CON-    *)
1660    (*          CERNING MESSAGE SEQUENCE TRANSMISSION. EACH    *)
1661    (*          CASE OPERATES ON AN ADAPTER'S NEXT TRANSMIT    *)
1662    (*          TIME AND IN TURN ON OVERALL SIMULATION TIME.   *)
1663    (*                                                         *)
1664    (*******************************************************)
1665
1666       VAR
1667         RA, RD, RS, I, J, K, L, KK, TA, TS, TD : INTEGER;
1668         A, FOUND : BOOLEAN ;
1669         TEMPR, TEMP, TIME, PROB, BITS, BLKCT : REAL;
1670
1671       BEGIN
1672        TIME := 0.0;
1673        KK := 0;
1674        TEMPR := 0.0;
1675        TEMP := 0.0;
1676        PROB := 0.0;
1677        BITS := 0.0;
1678        BLKCT := 0.0;
1679        A := FALSE;
1680        FOUND := FALSE;
1681
1682        TA := TMITTER.DNUM DIV 100;
1683        TD := (TMITTER.DNUM MOD 100) DIV 10;
1684        TS := TMITTER.DNUM MOD 10;
1685
1686        CASE COND OF
1687          CASE1:
1688
1689             BEGIN
1690              WITH ADAPTER[TA] DO
1691               WITH ADAPTER[TA].DEVICE[TD] DO
1692               WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1693               BEGIN
1694                 DFLAG[TD] := TRUE;
1695                 SFLAG[TS] := TRUE;
1696                 DELAY := 1.0E-06;
1697                 RETRY := 0;
1698                 INITIALTX := NEXTTX;
1699                 M1 := TRUE;
1700                 NEXTTX := NEXTTX + 64.0E-06;
1701                 TOTALATTEMPTS := TOTALATTEMPTS + 1;
1702
1703             END; (* CASE1 *)
```

```
1704                  END;
1705
1706          CASE2:
1707
1708            BEGIN
1709             WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1710              BEGIN
1711
1712                IF NOT ACOLLISION THEN
1713                 BEGIN
1714                  WHILE NOT FOUND DO
1715                   BEGIN
1716                    PROB := UNIFORM(U);
1717                    WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1718                     BEGIN
1719                      IF IDEST = COMMAD THEN
1720                       BEGIN
1721                        FOR J := 1 TO 4 DO
1722                         WITH ADAPTER[COMMAD].DEVICE[J] DO
1723                          IF NOT OPEN THEN
1724                           BEGIN
1725                            FOR K := 1 TO NUMOFSOURCES DO
1726                             WITH ADAPTER[COMMAD].DEVICE[J].SOURCE[K] DO
1727                              BEGIN
1728                               IF PROB < 0.5 THEN
1729                                BEGIN
1730                                 ADAPTER[TA].DEVICE[TD].SOURCE[TS].IDEST := DNUM;
1731                                 FOUND := TRUE;
1732                                END;
1733                              END;
1734                           END;
1735                       END
1736                      ELSE
1737                       FOUND := TRUE;
1738                     END;
1739                   END;
1740        IF IDEST <> 0 THEN
1741         BEGIN
1742          TRK2ATTEMPT := TRK2ATTEMPT + 1;
1743          RA := IDEST DIV 100;
1744          RD := (IDEST MOD 100) DIV 10;
1745          RS := IDEST MOD 10;
1746
1747          WITH ADAPTER[RA] DO
1748           BEGIN
1749            FOR L := 1 TO 4 DO
1750             IF DFLAG[L]  THEN
1751                A := TRUE;
1752           END;
1753         END
1754        ELSE
1755         BEGIN
1756          RA := DEST DIV 100;
1757          RD := (DEST MOD 100) DIV 10;
1758          RS := DEST MOD 10;
```

```
1759          WITH ADAPTER[RA] DO
1760           BEGIN
1761            FOR L := 1 TO 4 DO
1762             IF DFLAG[L] THEN
1763              A := TRUE;
1764           END;
1765         END;
1766
1767
1768        IF NOT A THEN
1769         BEGIN
1770          WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1771           WITH ADAPTER[RA].DEVICE[RD] DO
1772            WITH ADAPTER[RA] DO
1773            BEGIN
1774             DFLAG[RD] := TRUE;
1775             SFLAG[RS] := TRUE;
1776             FROMTX := TMITTER.DNUM;
1777             IF TRK1 THEN
1778              TEMPR := DISTFROMA1
1779             ELSE
1780              TEMPR := DISTFRTA1;
1781           END;
1782
1783        WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1784         WITH ADAPTER[TA] DO
1785          BEGIN
1786           IF TRK1 THEN
1787             TEMP := ABS(DISTFROMA1 - TEMPR)
1788           ELSE
1789             TEMP := ABS(DISTFRTA1 - TEMPR);
1790
1791           CARUP := NEXTTX;
1792           CARDWN := NEXTTX + 12.0E-06 + 2 * (TEMP * PROPAGATION);
1793           NEXTTX := NEXTTX + (CARDWN - CARUP) + 23.0E-06;
1794           M2 := TRUE;
1795           M1 := FALSE;
1796
1797           IF TRK1 THEN
1798            BEGIN
1799             TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP);
1800             BUSYTIME  := BUSYTIME + (CARDWN - CARUP);
1801             CONT1BITS := CONT1BITS + 600;
1802            END
1803           ELSE
1804            BEGIN
1805             TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
1806             BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1807             CONT2BITS := CONT2BITS + 600;
1808            END;
1809
1810           IF IDEST <> 0 THEN
1811             RX := IDEST;
1812           WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1813
```

```
1814            BEGIN
1815             BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1816             IF KK <> 0 THEN
1817              DEST := TMITTER.DEST;
1818            END;
1819          END;
1820        END
1821      ELSE
1822        BEGIN
1823         WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1824
1825            IF TRK1 THEN
1826             TEMPR := DISTFROMA1
1827            ELSE
1828             TEMPR := DISTFRTA1;
1829
1830             WITH ADAPTER[TA] DO
1831             WITH ADAPTER[TA].DEVICE[TD] DO
1832             WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1833             BEGIN
1834              IF TRK1 THEN
1835               TEMP := ABS(TEMPR - DISTFROMA1)
1836              ELSE
1837               TEMP := ABS(TEMPR - DISTFRTA1);
1838
1839              CARUP := NEXTTX;
1840              CARDWN := NEXTTX + 12.0E-06 + 2 * (TEMP * PROPAGATION);
1841
1842              IF DELAY < 128.0E-06 THEN
1843               BEGIN
1844                NEXTTX := NEXTTX + DELAY + ( CARDWN - CARUP);
1845                DELAY := 2 * DELAY;
1846               END
1847              ELSE
1848               BEGIN
1849                RETRY := RETRY + 1 ;
1850                IF RETRY > RETRYCT THEN
1851                  BEGIN
1852                   M1 := FALSE;
1853                   M2 := FALSE;
1854                   FINALTX := NEXTTX;
1855                   MESSDELAY := (FINALTX - INITIALTX) + MESSDELAY;
1856                   ABORTCT := ABORTCT + 1;
1857                   TOTALABORTS := TOTALABORTS + 1;
1858                   DFLAG[TD] := FALSE;
1859                   SFLAG[TS] := FALSE;
1860                   NEXTTX := NEXTTX + TXINTRVL;
1861                   DEST := 0;
1862                   IDEST := 0;
1863                  END
1864                ELSE
1865                 DELAY := 1.0E-06;
1866              END;
1867
1868
```

```
1869              IF TRK1 THEN
1870               BEGIN
1871                TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP);
1872                BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1873                CONT1BITS := CONT1BITS + 600;
1874               END
1875              ELSE
1876               BEGIN
1877                TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
1878                BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1879                CONT2BITS := CONT2BITS + 600;
1880               END;
1881             WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1882              BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1883
1884             END;
1885           END;
1886         END
1887        ELSE
1888        BEGIN
1889         CARUP := NEXTTX;
1890         CARDWN := NEXTTX + TIMEDIF;
1891         NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
1892         COLLCT := COLLCT + 1;
1893         COLLTIME := COLLTIME + TIMEDIF;
1894        END;
1895      END;
1896     END; (* CASE 2 *)
1897
1898
1899
1900     CASE3:
1901
1902
1903      BEGIN
1904       WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1905        BEGIN
1906         IF NOT ACCLLISION THEN
1907          BEGIN
1908           IF IDEST <> 0 THEN
1909            BEGIN
1910             RA := IDEST DIV 100;
1911             RD := (IDEST MOD 100) DIV 10;
1912             RS := IDEST MOD 10;
1913            END
1914           ELSE
1915            BEGIN
1916             RA := DEST DIV 100;
1917             RD := (DEST MOD 100) DIV 10;
1918             RS := DEST MOD 10;
1919             WRITELN(OUT, RA, RD, RS);
1920            END;
1921       WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1922        IF TRK1 THEN
1923         TEMPR := DISTFROMA1
```

```
1924        ELSE
1925         TEMPR := DISTFRTA1;
1926        IF TRK1 THEN
1927         TEMP := ABS(TEMPR - DISTFROMA1)
1928        ELSE
1929         TEMP := ABS(TEMPR - DISTFRTA1);
1930
1931        CARUP := NEXTTX;
1932        CARDWN := NEXTTX          + 96.5E-06 + 8 * (TEMP * PROPAGATION);
1933
1934        IF TRK1 THEN
1935         BEGIN
1936          CONT1BITS := CONT1BITS + 4313;
1937          DATA1BITS := DATA1BITS + 512;
1938          TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP)
1939         END
1940        ELSE
1941         BEGIN
1942          CONT2BITS := CONT2BITS + 4313;
1943          DATA2BITS := DATA2BITS + 512;
1944          TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
1945         END;
1946            BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1947
1948        WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1949         BUSYTIME := BUSYTIME + (CARDWN - CARUP);
1950
1951        M2 := FALSE;
1952        M3 := TRUE;
1953        NEXTTX := NEXTTX + (CARDWN - CARUP) + 64.0E-06;
1954      END
1955     ELSE
1956      BEGIN
1957       CARUP := NEXTTX;
1958       CARDWN := NEXTTX + TIMEDIF;
1959        NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
1960       COLLCT := COLLCT + 1;
1961       COLLTIME := COLLTIME + TIMEDIF;
1962      END;
1963      END;
1964    END; (* CASE 3 *)
1965
1966
1967     CASE4:
1968       BEGIN
1969        WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
1970         BEGIN
1971          IF NOT ACOLLISION THEN
1972            BEGIN
1973             IF IDEST <> 0 THEN
1974              BEGIN
1975               RA := IDEST DIV 100;
1976               RD := (IDEST MOD 100) DIV 10;
1977               RS := IDEST MOD 10;
1978              END
```

```
1979            ELSE
1980             BEGIN
1981              RA := DEST DIV 100;
1982              RD := (DEST MOD 100) DIV 10;
1983              RS := DEST MOD 10;
1984             END;
1985           WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
1986            IF TRK1 THEN
1987             TEMPR := DISTFROMA1
1988            ELSE
1989             TEMPR := DISTFRTA1;
1990
1991            IF TRK1 THEN
1992             TEMP := ABS(TEMPR - DISTFROMA1)
1993            ELSE
1994             TEMP := ABS(TEMPR - DISTFRTA1);
1995
1996            CARUP :=  NEXTTX;
1997            CARDWN := NEXTTX + 12.0E-06 + 2 * (TEMP * PROPAGATION);
1998
1999            IF TRK1 THEN
2000             BEGIN
2001              CONT1BITS := CONT1BITS + 600;
2002             TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP)
2003            END
2004           ELSE
2005            BEGIN
2006             CONT2BITS := CONT2BITS + 600;
2007             TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
2008            END;
2009            WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2010             BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2011
2012            M3 := FALSE;
2013            M4 := TRUE;
2014            NEXTTX := NEXTTX + (CARDWN - CARUP) + 29.0E-06;
2015           END
2016         ELSE
2017          BEGIN
2018           CARUP := NEXTTX;
2019           CARDWN := NEXTTX + TIMEDIF;
2020            NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
2021            COLLCT := COLLCT + 1;
2022            COLLTIME := COLLTIME + TIMEDIF;
2023          END;
2024         END;
2025       END; (* CASE 4 *)
2026
2027
2028     CASE5:
2029
2030       BEGIN
2031
2032        WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2033         BEGIN
```

```
2034          IF NOT ACOLLISION THEN
2035           BEGIN
2036            IF (IDEST = 0 ) AND (MB) THEN
2037              BLKCT := TBLK
2038            ELSE
2039              BLKCT := DATABLKCT;
2040
2041            IF (BLKCT <= 1) AND (BLKCT > BLKSENT) THEN
2042              BEGIN
2043               TIME := 84.32E-06 +((BLKCT * 16384)/ 50.0E+06);
2044               BLKSENT := BLKSENT + 1 ;
2045               BITS := BLKCT * 16384;
2046              END
2047            ELSE
2048              BEGIN
2049               IF (BLKCT > 1) AND (BLKCT > (BLKSENT + 1)) THEN
2050                 BEGIN
2051                  TIME := 412.0E-06;
2052                  BLKSENT := BLKSENT + 1;
2053                  BITS := 16384;
2054                 END
2055               ELSE
2056                 BEGIN
2057                  PROB := BLKCT - (BLKSENT - 1);
2058                  TIME := 84.32E-06 + ((PROB * 16384)/ 50.0E+06);
2059                  BITS := PROB * 16384;
2060                  BLKSENT := BLKSENT + PROB;
2061                END;
2062              END;
2063                IF IDEST <> 0 THEN
2064                 BEGIN
2065                  RA := IDEST DIV 100;
2066                  RD := (IDEST MOD 100) DIV 10;
2067                  RS := IDEST MOD 10;
2068                 END
2069                ELSE
2070                 BEGIN
2071                  RA := DEST DIV 100;
2072                  RD := (DEST MOD 100) DIV 10;
2073                  RS := DEST MOD 10;
2074                 END;
2075
2076           WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2077            IF TRK1 THEN
2078             TEMPR := DISTFROMA1
2079            ELSE
2080             TEMPR := DISTFRTA1;
2081
2082            IF TRK1 THEN
2083             TEMP := ABS(TEMPR - DISTFROMA1)
2084            ELSE
2085             TEMP := ABS(TEMPR - DISTFRTA1);
2086
2087            CARUP := NEXTTX;
2088            CARDWN := NEXTTX + TIME + 6 * ( TEMP * PROPAGATION);
```

```
2089
2090            IF TRK1 THEN
2091             BEGIN
2092             CONT1BITS := CONT1BITS + 4216;
2093             DATA1BITS := DATA1BITS + BITS;
2094             TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP)
2095             END
2096            ELSE
2097             BEGIN
2098             CONT2BITS := CONT2BITS + 4216;
2099             DATA2BITS := DATA2BITS + BITS;
2100             TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
2101             END;
2102
2103            BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2104            WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2105             BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2106
2107            M4 := FALSE;
2108            M5 := TRUE;
2109            NEXTTX := NEXTTX + (CARDWN - CARUP) + 34.0E-06;
2110          END
2111        ELSE
2112          BEGIN
2113           CARUP := NEXTTX ;
2114           CARDWN := NEXTTX + TIMEDIF;
2115             NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
2116             COLLCT := COLLCT + 1;
2117             COLLTIME := COLLTIME + TIMEDIF;
2118          END;
2119       END;
2120      END;  (* CASE 5 *)
2121
2122
2123
2124     CASE6:
2125
2126
2127      BEGIN
2128       WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2129        BEGIN
2130         IF NOT ACOLLISION THEN
2131          BEGIN
2132
2133           IF (IDEST = 0) AND (M8) THEN
2134             BLKCT := TBLK
2135           ELSE
2136             BLKCT := DATABLKCT;
2137
2138           IF IDEST <> 0 THEN
2139            BEGIN
2140             RA := IDEST DIV 100;
2141             RD := (IDEST MOD 100) DIV 10;
2142             RS := IDEST MOD 10;
2143            END
```

```
2144            ELSE
2145             BEGIN
2146              RA := DEST DIV 100;
2147              RD := (DEST MOD 100) DIV 10;
2148              RS := DEST MOD 10;
2149             END;
2150
2151           WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2152             IF TRK1 THEN
2153              TEMPR := DISTFROMA1
2154             ELSE
2155              TEMPR := DISTFRTA1;
2156
2157             IF TRK1 THEN
2158              TEMP := ABS(TEMPR - DISTFROMA1)
2159             ELSE
2160              TEMP := ABS(TEMPR - DISTFRTA1);
2161
2162             CARUP :=  NEXTTX;
2163             CARDWN := NEXTTX + 12.0E-06 + 2 * (TEMP * PROPAGATION);
2164
2165             IF TRK1 THEN
2166              BEGIN
2167               CONT1BITS := CONT1BITS + 600;
2168               TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP)
2169              END
2170             ELSE
2171              BEGIN
2172               CONT2BITS := CONT2BITS + 600;
2173               TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
2174              END;
2175
2176             BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2177             WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2178              BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2179             IF BLKSENT >= BLKCT THEN
2180              BEGIN
2181               M5 := FALSE;
2182               M6 := TRUE;
2183               NEXTTX := NEXTTX + (CARDWN - CARUP) + 20.0E-06;
2184              END
2185             ELSE
2186              BEGIN
2187               M4 := TRUE;
2188               M5 := FALSE;
2189               NEXTTX := NEXTTX + (CARDWN - CARUP) + 29.0E-06;
2190              END;
2191           END
2192           ELSE
2193            BEGIN
2194            CARUP := NEXTTX;
2195            CARDWN := NEXTTX + TIMEDIF;
2196            NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
2197            COLLCT := COLLCT + 1;
2198            COLLTIME := COLLTIME + TIMEDIF;
```

```
2199            END;
2200          END;
2201        END; (* CASE 6 *)
2202
2203
2204      CASE7:
2205
2206
2207       BEGIN
2208        WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2209         BEGIN
2210          IF NOT ACOLLISION THEN
2211           BEGIN
2212             IF IDEST <> 0 THEN
2213              BEGIN
2214               RA := IDEST DIV 100;
2215               RD := (IDEST MOD 100) DIV 10;
2216               RS := IDEST MOD 10;
2217              END
2218             ELSE
2219              BEGIN
2220               RA := DEST DIV 100;
2221               RD := ( DEST MOD 100) DIV 10;
2222               RS := DEST MOD 10;
2223              END;
2224
2225          WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2226           IF TRK1 THEN
2227            TEMPR := DISTFROMA1
2228           ELSE
2229            TEMPR := DISTFRTA1;
2230
2231           IF TRK1 THEN
2232            TEMP := ABS(TEMPR - DISTFROMA1)
2233           ELSE
2234            TEMP := ABS(TEMPR - DISTFRTA1);
2235
2236           CARUP :=  NEXTTX;
2237           CARDWN := NEXTTX + 12.0E-06 + 2 * (TEMP * PROPAGATION);
2238
2239           IF TRK1 THEN
2240            BEGIN
2241              SEQ1TALLY := SEQ1TALLY + 1;
2242              TR1ACTIVE := TR1ACTIVE + (CARDWN - CARUP);
2243              CONT1BITS := CONT1BITS + 600;
2244            END
2245           ELSE
2246            BEGIN
2247              SEQ2TALLY := SEQ2TALLY + 1 ;
2248              TR2ACTIVE := TR2ACTIVE + (CARDWN - CARUP);
2249              CONT2BITS := CONT2BITS + 600;
2250            END;
2251
2252          SEQTALLY := SEQTALLY + 1;
2253          SEQCT := SEQCT + 1;
```

```
2254          FINALTX := CARDWN + 20.0E-06;
2255          MESSDELAY := MESSDELAY + (FINALTX - INITIALTX);
2256          M6 := FALSE;
2257          WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2258           BUSYTIME := BUSYTIME + (CARDWN - CARUP);
2259
2260          IF (IDEST = 0 ) AND (NOT M8) THEN
2261          WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2262           WITH ADAPTER[TA].DEVICE[TD] DO
2263            WITH ADAPTER[TA] DO
2264            BEGIN
2265             DFLAG[TD] := FALSE;
2266             SFLAG[TS] := FALSE;
2267             NEXTTX := NEXTTX + TXINTRVL;
2268             DEST := 0;
2269             BLKSENT := 0.0;
2270             WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2271              WITH ADAPTER[RA].DEVICE[RD] DO
2272               WITH ADAPTER[RA] DO
2273               BEGIN
2274                DFLAG[RD] := FALSE;
2275                SFLAG[RS] := FALSE;
2276                FROMTX := 0;
2277                RSEQCT := RSEQCT + 1;
2278               END;
2279            END;
2280
2281          IF (IDEST <> 0 ) AND (NOT M8 ) THEN
2282            BEGIN
2283             WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2284              WITH ADAPTER[RA].DEVICE[RD] DO
2285               WITH ADAPTER[RA] DO
2286               BEGIN
2287                TBLK := ADAPTER[TA].DEVICE[TD].SOURCE[TS].DATABLKCT;
2288                DFLAG[RD] := FALSE;
2289                SFLAG[RS] := FALSE;
2290                M8 := TRUE;
2291                NEXTTX := ADAPTER[TA].DEVICE[TD].SOURCE[TS].FINALTX;
2292                FROMTX := 0;
2293                RSEQCT := RSEQCT + 1 ;
2294               END;
2295               WITH ADAPTER[TA] DO
2296                WITH ADAPTER[TA].DEVICE[TD] DO
2297                 WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2298                 BEGIN
2299                  DFLAG[TD] := FALSE;
2300                  SFLAG[TS] := FALSE;
2301                  NEXTTX := NEXTTX + TXINTRVL;
2302                  DEST := 0;
2303                  IDEST := 0;
2304                  BLKSENT := 0.0;
2305                 END;
2306             END;
2307
2308            IF (IDEST = 0) AND (M8) THEN
```

```
2309              WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2310               WITH ADAPTER[TA].DEVICE[TD] DO
2311                WITH ADAPTER[TA] DO
2312                BEGIN
2313                 TRK2TALLY := TRK2TALLY + 1;
2314                 MB := FALSE;
2315                 NEXTTX := FINALTX + TXINTRVL;
2316                 DFLAG[TD] := FALSE;
2317                 SFLAG[TS] := FALSE;
2318                 DEST := 0;
2319                 FROMTX := 0;
2320                 TBLK := 0.0;
2321                 BLKSENT := 0.0;
2322
2323                 WITH ADAPTER[RA].DEVICE[RD].SOURCE[RS] DO
2324                  WITH ADAPTER[RA].DEVICE[RD] DO
2325                   WITH ADAPTER[RA] DO
2326                   BEGIN
2327                    DFLAG[RD] := FALSE;
2328                    SFLAG[RS] := FALSE;
2329                    FROMTX := 0;
2330                    RSEQCT := RSEQCT + 1;
2331                   END;
2332               END;
2333             END
2334            ELSE
2335             BEGIN
2336              CARUP := NEXTTX;
2337              CARDWN := NEXTTX + TIMEDIF;
2338              NEXTTX := NEXTTX + FIXEDDELAY + PRIDELAY;
2339              COLLCT := COLLCT + 1;
2340              COLLTIME := COLLTIME + TIMEDIF;
2341             END;
2342           END;
2343         END;  (* CASE 7 *)
2344       END;    (* CASE COND OF *)
2345      END;     (* UPDATE *)
2346
2347
2348
2349
2350  (*************************************************************************)
2351  (*                                                                       *)
2352           PROCEDURE PRINTSTATS;
2353  (*                                                                       *)
2354  (*************************************************************************)
2355
2356    VAR
2357      I, J, K          : INTEGER;
2358      PER1ACTIVE, PER2ACTIVE, PERTACTIVE      : REAL;
2359
2360    BEGIN
2361
2362      FOR I := 1 TO 10 DO  WRITELN(OUTPUT);
2363
```

```
2364        IF TRUNKS THEN
2365         BEGIN
2366          PER1ACTIVE :=(TR1ACTIVE / CURRENTTIME) * 100;
2367          PER2ACTIVE :=(TR2ACTIVE / CURRENTTIME) * 100;
2368          PERTACTIVE :=(TIMEACTIVE / CURRENTTIME) * 100;
2369         END
2370        ELSE
2371         PERTACTIVE :=(TIMEACTIVE / CURRENTTIME) * 100;
2372
2373        FOR I := 1 TO 15 DO WRITELN(AUXOUT);
2374
2375      WRITELN(AUXOUT,' ':25, '*** END OF RUN NETWORK STATISTICS ***');
2376      WRITELN(AUXOUT);
2377      WRITELN(AUXOUT,' ':20, 'CURRENT TIME : ', CURRENTTIME:12:4,' SECS');
2378      WRITELN(AUXOUT);
2379      WRITELN(AUXOUT,' ':20, 'SUCCESSFUL SEQUENCE TRANSMISSIONS          : ',
2380              SEQTALLY: 5);
2381      IF TRUNKS THEN
2382      BEGIN
2383      WRITELN(AUXOUT,' ':20, 'SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK 1  : ',
2384              SEQ1TALLY: 5);
2385      WRITELN(AUXOUT,' ':20, 'SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK 2  : ',
2386              SEQ2TALLY :5);
2387      END;
2388      WRITELN(AUXOUT,' ':20, 'COLLISIONS   (FRAMES)                      : ',
2389              COLLTALLY :5);
2390      WRITELN(AUXOUT,' ':20, 'WAITS        (FRAMES)                      : ',
2391              WAITTALLY :5);
2392      WRITELN(AUXOUT,' ':20, 'TOTAL ATTEMPTS (SEQUENCES)                 : ',
2393              TOTALATTEMPTS:5);
2394      WRITELN(AUXOUT,' ':20, 'TOTAL ABORTS                               : ',
2395              TOTALABORTS: 5);
2396      IF TRUNKS THEN
2397      BEGIN
2398      WRITELN(AUXOUT,' ':20, 'ATTEMPTED TRUNK-TRUNK TRANSMISSIONS        : ',
2399              TRK2ATTEMPT: 5);
2400      WRITELN(AUXOUT,' ':20, 'SUCCESSFUL TRUNK-TRUNK TRANSMISSIONS       : ',
2401              TRK2TALLY :5);
2402      END;
2403      WRITELN(AUXOUT);
2404      IF TRUNKS THEN
2405      BEGIN
2406      WRITELN(AUXOUT,' ':20, 'TRUNK 1 ACTIVE TIME                        : ',
2407              TR1ACTIVE :10 :7,' SECS');
2408      WRITELN(AUXOUT,' ':20, 'TRUNK 2 ACTIVE TIME                        : ',
2409              TR2ACTIVE :10 :7,' SECS');
2410      WRITELN(AUXOUT,' ':20, 'TOTAL TRUNK ACTIVE TIME                    : ',
2411              TIMEACTIVE :10 :7,' SECS');
2412      WRITELN(AUXOUT,' ':20, '% TRUNK 1 ACTIVE TIME                      : ',
2413              PER1ACTIVE: 6: 3,' %');
2414      WRITELN(AUXOUT,' ':20, '% TRUNK 2 ACTIVE TIME                      : ',
2415              PER2ACTIVE: 6:3,' %');
2416      WRITELN(AUXOUT,' ':20, '% TOTAL ACTIVE TIME                        : ',
2417              PERTACTIVE: 6:3,' %');
2418      END
```

```
2419      ELSE
2420       BEGIN
2421       WRITELN(AUXOUT,' ':20, 'TOTAL TRUNK ACTIVE TIME                    : ',
2422              TIMEACTIVE: 10:7,' SECS');
2423       WRITELN(AUXOUT,' ':20, '% TOTAL ACTIVE TIME                        : ',
2424              PERTACTIVE: 6:3  ,' %');
2425       END;
2426       WRITELN(AUXOUT);
2427
2428       IF NOT TRUNKS THEN
2429       BEGIN
2430       WRITELN(AUXOUT,' ':20, 'CONTROL BYTES TRANSMITTED                  : ',
2431              CONT1BITS / 8.0E+06:12,' MBYTES');
2432       WRITELN(AUXOUT,' ':20, 'DATA BYTES TRANSMITTED                     : ',
2433              DATA1BITS / 8.0E+06 :12,' MBYTES');
2434       END
2435      ELSE
2436       BEGIN
2437       WRITELN(AUXOUT,' ':20, 'CONTROL BYTES TRANSMITTED - TRUNK 1        : ',
2438              CONT1BITS / 8.0E+06 :12, ' MBYTES');
2439       WRITELN(AUXOUT,' ':20, 'DATA BYTES TRANSMITTED    - TRUNK 1        : ',
2440              DATA1BITS / 8.0E+06 :12, ' MBYTES');
2441       WRITELN(AUXOUT,' ':20, 'CONTROL BYTES TRANSMITTED - TRUNK 2        : ',
2442              CONT2BITS / 8.0E+06 :12, ' MBYTES');
2443       WRITELN(AUXOUT,' ':20, 'DATA BYTES TRANSMITTED    - TRUNK 2        : ',
2444              DATA2BITS / 8.0E+06 :12, ' MBYTES');
2445       END;
2446       WRITELN(AUXOUT,' ':20, 'TOTAL BYTES TRANSMITTED                    : ',
2447              BITSTX / 8.0E+06 :12,' MBYTES');
2448       WRITELN(AUXOUT);
2449       WRITELN(AUXOUT,' ':20, 'TOTAL OFFERED LOAD                        : ',
2450              (OFFLOAD * CURRENTTIME)/ 8.0E+06 :12,' MBYTES');
2451
2452       WRITELN(' ':15, ' *** END OF RUN NETWORK STATISTICS ***');
2453       WRITELN;
2454       WRITELN(' ':10, ' CURRENT TIME   :  ', CURRENTTIME: 12: 4, 'SECS');
2455       WRITELN;
2456       WRITELN(' ':10, ' SUCCESSFUL SEQUENCE TRANSMISSIONS                : ',
2457              SEQTALLY: 5);
2458       IF TRUNKS THEN
2459       BEGIN
2460       WRITELN(' ':10, ' SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK1         : ',
2461              SEQ1TALLY: 5 );
2462       WRITELN(' ':10, ' SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK2         : ',
2463              SEQ2TALLY: 5 );
2464       END;
2465       WRITELN(' ':10, ' COLLISIONS        (FRAMES)                       : ',
2466              COLLTALLY: 5);
2467       WRITELN(' ':10, ' WAITS             (FRAMES)                       : ',
2468              WAITTALLY: 5);
2469       WRITELN(' ':10, ' TOTAL ATTEMPTS    (SEQUENCES)                    : ',
2470              TOTALATTEMPTS: 5 );
2471       WRITELN(' ':10, ' TOTAL ABORTS                                     : ',
2472              TOTALABORTS: 5);
2473       IF TRUNKS THEN
```

```
2474      BEGIN
2475      WRITELN(' ':10, ' ATTEMPTED TRUNK-TRUNK TRANSMISSIONS              : ',
2476              TRK2ATTEMPT: 5);
2477      WRITELN(' ':10, ' SUCCESSFUL TRUNK-TRUNK TRANSMISSIONS             : ',
2478              TRK2TALLY: 5);
2479      WRITELN;
2480      WRITELN(' ':10, ' TRUNK 1 ACTIVE TIME                              : ',
2481              TR1ACTIVE: 10: 7,' SECS');
2482      WRITELN(' ':10, ' TRUNK 2 ACTIVE TIME                              : ',
2483              TR2ACTIVE: 10: 7,' SECS');
2484      WRITELN(' ':10, ' TOTAL TRUNK ACTIVE TIME                          : ',
2485              TIMEACTIVE: 10: 7,' SECS');
2486      WRITELN(' ':10, ' % TRUNK 1 ACTIVE TIME                            : ',
2487              PER1ACTIVE:  6: 3, ' %');
2488      WRITELN(' ':10, ' % TRUNK 2 ACTIVE TIME                            : ',
2489              PER2ACTIVE:  6: 3, ' %');
2490      WRITELN(' ':10, ' % TOTAL TRUNK ACTIVE TIME                        : ',
2491              PERTACTIVE:  6: 3, ' %');
2492       END
2493     ELSE
2494      BEGIN
2495      WRITELN(' ':10, ' TOTAL TRUNK ACTIVE TIME                          : ',
2496              TIMEACTIVE: 10: 7,' SECS');
2497      WRITELN(' ':10, ' % TOTAL TRUNK ACTIVE TIME                        : ',
2498              PERTACTIVE: 6: 3 , ' %');
2499      END;
2500
2501      WRITELN;
2502
2503      IF NOT TRUNKS THEN
2504      BEGIN
2505      WRITELN(' ':10, ' CONTROL BYTES TRANSMITTED                        : ',
2506            CONT1BITS / 8.0E+06: 12,' MBYTES');
2507      WRITELN(' ':10, ' DATA BYTES TRANSMITTED                           : ',
2508            DATA1BITS / 8.0E+06: 12,' MBYTES');
2509      END
2510     ELSE
2511      BEGIN
2512      WRITELN(' ':10, 'CONTROL BYTES TRANSMITTED     TRUNK 1             : ',
2513            CONT1BITS / 8.0E+06: 12, ' MBYTES');
2514      WRITELN(' ':10, 'DATA BYTES TRANSMITTED        TRUNK 1             : ',
2515            DATA1BITS / 8.0E+06: 12, ' MBYTES');
2516      WRITELN(' ':10, 'CONTROL BYTES TRANSMITTED     TRUNK 2             : ',
2517            CONT2BITS / 8.0E+06: 12, ' MBYTES');
2518      WRITELN(' ':10, 'DATA BYTES TRANSMITTED        TRUNK 2             : ',
2519            DATA2BITS / 8.0E+06: 12, ' MBYTES');
2520      END;
2521      WRITELN(' ':10, ' TOTAL BYTES TRANSMITTED                          : ',
2522             BITSTX / 8.0E+06: 12,' MBYTES');
2523      WRITELN;
2524      WRITELN(' ':10, ' TOTAL OFFERED LOAD                               : ',
2525             (OFFLOAD * CURRENTTIME)/ 8.0E+06: 12, ' MBYTES');
2526
2527      END;
2528
```

```
2529
2530    (*************************************************************)
2531    (*                                                           *)
2532            PROCEDURE ACTIVITYSUMMARY;
2533    (*                                                           *)
2534    (*************************************************************)
2535
2536     VAR
2537        I, J, K       : INTEGER;
2538        MDELAY        : REAL;
2539
2540
2541     BEGIN
2542      WRITELN(AUXOUT);
2543      WRITELN(AUXOUT);
2544      WRITELN(AUXOUT);
2545      WRITELN(AUXOUT);
2546      WRITELN(AUXOUT,' ':45, 'DEVICE ACTIVITY SUMMARIES');
2547      WRITELN(AUXOUT,' ':52,     '(SECONDS)');
2548      WRITELN(AUXOUT);
2549      WRITE  (AUXOUT,' ':2, 'ADP DEV SOURCE');
2550      WRITE  (AUXOUT,' ':3, 'TIME',' ':9, 'TIME',' ':10);
2551      WRITE  (AUXOUT,'TIME IN',' ':3, 'AVG MESSAGE',' ':5, 'ABORT');
2552      WRITE  (AUXOUT,' ':9, 'TRANSMISSION',' ':2, 'RECEPTION',' ':4,'WAIT');
2553      WRITELN(AUXOUT,' ':3,'COLLISION');
2554      WRITE  (AUXOUT,'  #    #    #',' ':4,'ACTIVE',' ':7,'WAITING',' ':7);
2555      WRITE  (AUXOUT, 'COLLISIONS',' ':4, 'DELAY',' ':10, 'COUNT',' ':12);
2556      WRITE  (AUXOUT, 'COUNT',' ':8, 'COUNT',' ':7, 'COUNT',' ':4);
2557      WRITELN(AUXOUT, 'COUNT');
2558      WRITELN(AUXOUT);
2559
2560      FOR I := 1 TO NUMOFADAPTERS DO
2561       FOR J := 1 TO 4 DO
2562        WITH ADAPTER[I].DEVICE[J] DO
2563         IF NOT OPEN THEN
2564          FOR K := 1 TO NUMOFSOURCES DO
2565           WITH ADAPTER[I].DEVICE[J].SOURCE[K] DO
2566            BEGIN
2567             MDELAY := 0.0;
2568             IF(MESSDELAY <> 0.0) AND (SEQCT <> 0 ) THEN
2569               MDELAY := MESSDELAY / SEQCT;
2570             WRITE  (AUXOUT, I: 4, J: 4, K: 4, BUSYTIME: 12: 4, WAITTIME:13:4);
2571             WRITE  (AUXOUT,' ':7, COLLTIME: 5,' ':5,MDELAY:5,ABORTCT:10);
2572             WRITE  (AUXOUT, SEQCT: 16, RSEQCT: 14, WAITCT:11);
2573             WRITELN(AUXOUT,COLLCT: 9);
2574             WRITELN(AUXOUT, ID);
2575             WRITELN(AUXOUT);
2576            END;
2577       END;
2578    [*********************************************************]
2579    [*********************************************************]
2580    [            M A I N     P R O G R A M                    ]
2581    [*********************************************************]
2582
2583    BEGIN [ MAIN PROGRAM ]
```

```
2584
2585        { NETWORK COLD START CONDITIONS }
2586
2587        INITIALIZE;
2588
2589        CHARACTERIZENETWORK;
2590
2591        PRINTNETDESCRIPTION;
2592
2593       REPEAT
2594          CARDWN := 0.0; CARUP := 0.0;
2595          FINDNEXT(TMITTER);
2596          CURRENTTIME := TMITTER.NEXTTX;
2597          TA := TMITTER.DNUM DIV 100;
2598          TD := (TMITTER.DNUM MOD 100) DIV 10;
2599          TS := TMITTER.DNUM MOD 10;
2600          IF TMITTER.DEST = 0 THEN
2601           PICKA (TMITTER);
2602        WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2603         BEGIN
2604
2605           IF TA < COMMAD THEN
2606             TRK1 := TRUE
2607           ELSE
2608             TRK1 := FALSE;
2609
2610           IF TA = COMMAD THEN
2611             IF ((DEST DIV 100) < COMMAD ) THEN
2612              TRK1 := TRUE
2613             ELSE
2614              TRK1 := FALSE;
2615        END;
2616         WITH ADAPTER[TA] DO
2617          IF TRK1 THEN
2618           BEGIN
2619            CARUP := T1;
2620            CARDWN := T2;
2621            TOTALDELAY := TOT1DELAY;
2622            PRIDELAY   := PRI1DELAY;
2623            ENDDELAY   := END1DELAY;
2624            FIXEDDELAY := FIX1DELAY;
2625           END
2626        ELSE
2627           BEGIN
2628            CARUP := T3;
2629            CARDWN := T4;
2630            TOTALDELAY := TOT2DELAY;
2631            PRIDELAY   := PRI2DELAY;
2632            ENDDELAY   := END2DELAY;
2633            FIXEDDELAY := FIX2DELAY;
2634           END;
2635
2636        IF NOT RESERVED THEN
2637          WITH ADAPTER[TA].DEVICE[TD].SOURCE[TS] DO
2638           BEGIN
```

```
2639              IF M1 OR M2 OR M3 OR M4 OR M5 OR M6 THEN
2640                 BEGIN
2641         IF (NEXTTX >= CARDWN + TOTALDELAY) OR (NEXTTX = CARDWN + FIXEDDELAY
2642             + PRIDELAY ) THEN
2643           BEGIN
2644           IF M1 THEN
2645            UPDATE(CONDITION[CASE2]) ELSE
2646           IF M2 THEN
2647            UPDATE(CONDITION[CASE3]) ELSE
2648           IF M3 THEN
2649            UPDATE(CONDITION[CASE4]) ELSE
2650           IF M4 THEN
2651            UPDATE(CONDITION[CASE5]) ELSE
2652           IF M5 THEN
2653            UPDATE(CONDITION[CASE6]) ELSE
2654           IF M6 THEN
2655            UPDATE(CONDITION[CASE7])
2656         END
2657        ELSE
2658           BEGIN
2659           IF (NEXTTX <= CARDWN + FIXEDDELAY + PRIDELAY) THEN
2660                     NEXTTX := CARDWN + FIXEDDELAY + PRIDELAY;
2661           IF (NEXTTX > CARDWN + FIXEDDELAY + PRIDELAY ) THEN
2662                     NEXTTX := CARDWN + TOTALDELAY;
2663           END;
2664
2665        IF (NEXTTX < CARDWN) AND ((M1) OR (M2) OR (M3) OR (M4) OR (M5)
2666           OR (M6)) THEN
2667                     NEXTTX := CARDWN + FIXEDDELAY + PRIDELAY;
2668
2669           END
2670        ELSE
2671           UPDATE(CONDITION[CASE1]);
2672        END;
2673
2674         IF TRK1 THEN
2675          BEGIN
2676           T1 := CARUP;
2677           T2 := CARDWN;
2678          END
2679         ELSE
2680          BEGIN
2681           T3 := CARUP;
2682           T4 := CARDWN;
2683          END;
2684      UNTIL ( CURRENTTIME >= MAXTIME) OR ( SEQTALLY >= MAXSEQTX );
2685
2686         TIMEACTIVE := TR1ACTIVE + TR2ACTIVE;
2687         BITSTX := CONT1BITS + DATA1BITS + CONT2BITS + DATA2BITS;
2688
2689         PRINTSTATS;
2690         ACTIVITYSUMMARY;
2691
2692         WRITELN(OUTPUT);
2693         WRITELN('END OF RUN');
```

```
 2694       END. (* MAIN PROGRAM *)

COMPILATION COMPLETE :       NO ERRORS REPORTED



OPTIONS REQUESTED :-

   GENERATE EXTENDED ADDRESSING CODE ( X OPTION )

OBJECT PROGRAM        =    24563 WORDS

COMPILATION TIME      =    40473 MILLISECONDS

SOURCE PROGRAM        =     2694 LINES

COMPILATION RATE      =       67 LINES PER SECOND

===>      $VU.R
===>      SP S+20
===>      MO NX
===>      BEGIN
1                  PROGRAM NAME:  2028EING*XE
0                        LIBRARY INFORMATION
                           0000SYST*SAULIB
0                        MODULE INFORMATION
         R  0000000  0000000                                  P$IGVS
         R  0000002  0026353                                  P$MAIN
            0026354  0047373
            0047374  0057765
         R  0057766  0061733      0070000  0070102          P$REAR
         R  0061734  0066446      0070104  007045o          LIBERY
         R  0066450  0066557      0070460  0070460          P$STEQ
                                  0070462  007106C          *TEMP*
                                  0071062  0071161          *DBUG*
    S                             0102000  0102000          P$SCOM
```

APPENDIX II

Summary of ISO-OSI Model of Distributed Networks

Layer	Definition
1. Physical	Concerned with transmission of unstructured bit stream over physical link; involves such parameters as signal voltage swing and bit duration; deals with the mechanical, electrical, and procedureal characteristics to establish, maintain, and deactivate the physical link. (RS-232-C, RS-449, X.29)
2. Data Link	Converts an unreliable transmission channel into a reliable one; sends blocks of data (frames) with checksum; uses error detection and frame aknowledgement (HDLC, SDLC, BiSync)
3. Network	Transmits packets of data through a network; packets may be independent (datagram) or traverse a preestablished network connection (virtual circuit); responsible for routing and congestion control. (X.25, layer 3)
4. Transport	Provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control
5. Session	Provides means of establishing, managing, and terminating connection (session) between two processes; may provide checkpoint and restart service, quarantine service
6. Presentation	Performs generally useful transformations on data to provide a standardized application interface and to provide common communications services; i.e., encryption, text compression, reformatting
7. Application	Provides services to the users of OSI environment; i.e., transaction server, file transfer protocol, network management

This summary taken from, **Local Networks, An Introduction,** by William Stallings [STALL84].

APPENDIX III
Simulation Software Module Descriptions

1. Sim1.

Sim1 is the top level procedure of the simulation program. It controls simulation activity and flow in the simulation environment. Referring to Figure 4.2, the first three blocks perform variable initialization, define the simulation network configuration, and prints a description of the network in the auxiliary file Auxout. The loop indicated is the actual simulation activity (described in subsequent sections). Sim1 will print compiled overall network statistics to the system default output device and to the auxiliary file Auxout, by invoking Printstats. The procedure Activitysummary prints a breakdown of the network statistics by device, to the auxiliary file (Auxout).

2. Initialize

This procedure is used to initialize certain global variables which need to be specified for each execution of the program. Table III.1, is a listing of the interactive queries posed by this procedure, along with any constraints involved. The remaining variables which do not require user-intervention are also initialized in this section.

Table III.1 Interactive Queries Posed by INITIALIZE

Query:	**Is there more than one Trunk ? Y)es or N)o**
Response:	Y or N, if answered yes then the flag Trunks is set true, otherwise, Trunks is set false.
Example:	Y
Variable:	Trunks = True
Query:	**Identification heading for run:**
Response:	Valid responses are alphanumeric strings with length less than 50 characters. Heading appears on each output page.
Example:	Simulation Configuration Test
Variable:	Heading
Query:	**Maximum run time in seconds**
Response:	Valid responses are real positive numbers which represent the time a simulation run will execute
Example:	40
Variable:	Maxtime
Query:	**Maximum Successful Sequence Transmissions ?**
Response:	Valid responses are real positive numbers which represent the number of successful message-with-data sequences will be transferred over the entire network simulation ends when Maxseqtx or Maxtime is exceeded
Example:	100000
Variable:	Maxseqtx
Query:	**Seed for Random Number Generator**
Response:	Integer number. For best results, the number specified should be an odd, positive number with at least 5 digits and not divisible by 5 or 7.
Example:	16227
Variable:	U

The following queries will be posed if Trunks is True.

Query:	**Was the last Run a Single Trunk Configuration Y)es or N)o**
Response:	Y or N, if answered yes, Inputmode = interactive, thus forcing redefinition of the network. If answered no then the following query will be posed.
Example:	Y

Variable: Inputmode

Query: **Input Mode: I)nteractive or F)ile ?**
Response: This query will be posed only if the last run was a dual trunk configuration, this is indicated by answering no to the above query
If answered F, the network configuration used will be read out of the Dfile. If answered I, then the network will have be redefined by the user
Example: F
Variable Inputmode

The following queries will be posed if Trunks = False

Query: **Was the last run a Dual Trunk configuration Y)es or N)o**
Response: As before, yes will force redefinition of the network and no will cause the next query to be posed.
Example: N
Variable: Inputmode

Query: **Input Mode: I)nteractive or F)ile ?**
Response: As before, answering no to the above query will allow this query to be posed. The effect of this query is the same as above
Example: I
Variable: Inputmode

3. CharacterizeNetwork

Procedure CharacterizeNetwork establishes the experimental framework of the simulation. The individual adapters, devices, and sources are defined. There are two ways to establish an experimental model, interactively, or from an auxiliary file (Dfile). The method employed is determined by the value of Inputmode, defined by procedure Initialize. If Inputmode is equal to interactive, then the system description will be provided by the user interactively. A sub-procedure RewriteDfile will be called to store this interactive system description in the Dfile. If Inputmode equals file, then the system description used is the one stored in the auxiliary file (Dfile), implying a previous run where Dfile was created interactively. A detailed listing of the interactive queries posed by CharacterizeNetwork is presented in Table III.2.

4. PrintnetDescription

The purpose of this procedure is to print a detailed description of the system being modeled in the auxiliary file Auxout.

5. FindNext(Tmitter)

As previously stated this model is a discrete event simulation. The event which drives the simulation is an adapter's next scheduled transmit time. The

Table III.2 Queries Posed by CHARACTERIZENETWORK

Query:	**Length of Trunks in feet (used for end delay)** **Trunk 1:** **Trunk 2: (appears for dual trunk)**
Response:	Positive real numbers
Variable:	Trunklength[I]
Query:	**Number of Adapters in Network:**
Response:	Integer number in range 1 to Maxnumadanters
Variable:	Numofadapters

In the Dual Trunk case the following query will be made

Query:	**Which Adapter is Common to both Trunks ?**
Response	Integer number in range 1 to Maxnumadapters
Variable:	Commad
Query:	**Adapter #i: Distance in ft. from priority 1 adapter on Trunk 1**
Response:	Positive real numbers
Variable:	Adapter[i].distfromA1
Query:	**Adapter Priority: Give in integer form,i.e., n = 1,2,....**
Response:	Positive integer numbers
Variable:	Adapterp[i].P1
Query:	**Adapter Retry Count: This must be a unique number for each adapter, in the range 0-64**
Response:	Positive integer numbers
Variable:	Adapterp[i].retryct

The following two queries are made for the common adapter two both trunks

Query:	**Distance in ft. from priority 1 adapter on Trunk 2**
Response:	Positive real numbers
Variable:	Adapter[Commad].DistfrtA1
Query:	**Adapter Priority: Give in integer form,i.e., n = 1,2,.....**
Response:	Positive real numbers
Variable:	Adapter[Commad].P2
Query:	**Adapter[i].Device[j] description Device status (Open or Closed)**
Response:	O or C. C if Adapter[i].Device[j] has an active, network device con-

nected to it. 0 otherwise.
Variable: Adapter[i].Device[j].Open

Query: **Device ID (<= 24 Char)**
Response: Alphanumeric string
Variable: Adapter[i].Device[j].Id

Query: **Device to Adapter I/O rate (Bytes/sec)**
Response: Positive real number representing the speed of the device I/O port.
Variable: Adapter[i].Device[j].tferrate

Query: **Number of Data Sources in Device[j]:**
Response: Integer number representing the number of data sources supplying the device with data
Variable: Adapter[i].Device[j].Numofsources

Query: **Data Source[k] ID:**
Response: Alphanumeric string with length <= 24 characters
Variable: Adapter[i].Device[j].Source[k].Id

Query: **Source[k] Data gen rate (Bytes/sec)**
Response: Real number, representing the rate at which data arrives at device.
Variable: Adapter[i].Device[j].Source[k].Genrate

Query: **Buffer size in bytes**
Response: Integer number, size of buffer filled in device before network transmission is requested.
Variable: Adapter[i].Device[j].Source[k].Buffersize

Query: **Receiver Id code ? (0 to end list)**
Response: 3 digit integer number, representing Adapter#,Device#,and Source#
Variable: Adapter[i].Device[j].Source[k].Id

Query: **Probability ?**
Response: Positive real number 0 < number <= 1 representing the relative probability that the above receiver will be transmitted to by this adapter[i].device[j].source[k]
Variable: Adapter[i].Device[j].Source[k].Prob

procedure FindNext scans all adapters and picks the one with the smallest next time to transmit, compared to (Currenttime) simulation time. After selecting one adapter, it will then determine if this adapter is reserved for a reception of a message sequence from another adapter. If the adapter is reserved, then it is forced to wait, if not reserved then it is returned to the main program as the current transmitter.

6. PickA(Tmitter)

This procedure is used to pick a receiver for a transmitting adapter. The current transmitter's attributes are passed to this procedure, and an appropriate receiver is selected based on its list of possible receivers and the relative probability of the adapter being a receiver for the transmitter. This procedure also determines whether a transmitter is attempting to transmit to a different trunk. If attempting a dual trunk communication, the transmitter is assigned an intermediate receiver which is common to both trunks.

7. ACollision

The purpose of the boolean function ACollision, is to determine whether or not the current transmitter has unrestricted use of the simulated transmission medium. If another device attempts to transmit during a time period of equal to, or less than , twice the

propagation time between the two, a collision condition will be declared. Thus, ACollision scans the network devices comparing their next transmit time to the current transmitter's next transmit time and determines whether or not a collision has occurred. In dual trunk simulations the function ACollision will scan which ever trunk is being used by the current transmitter. If a collision is declared, then collision statistics are updated, and both the current transmitter's and colliding adapter's next transmission times are updated.

8. Uniform

The Uniform procedure functions as a Random Number Generator. It generates pseudo-random numbers in the range 0 < number < 1. This procedure is used in various sections of the simulation in order to inject some randomness in the selection of receivers and in varying some time parameters.

9. Update

Update is the procedure in which the bulk of the HYPERchannel protocol is modeled. As previously stated, the simulation is driven by a discrete event. This type of simulation is often referred to as activity scanning, or event scanning. The event which drives the simulation is a transmit request by an adapter. Each device is assumed to possess certain attributes

determined from observations of the real system. As an example of this, if a device receives data from an off-net source to transmit over the network at an average rate of 15 Kilobytes per second, and stores this data in a 2 Kbyte buffer, it will request a network transmission every 0.1365 seconds. This then serves to generate transmission requests for the simulation and in turn, becomes the event which drives the simulation clock.

This procedure consists of seven cases, each of which updates the current transmitter according to where it is in a message-with-data sequence transmission. The time periods involved are illustrated in Figure 4.1, where the individual cases are indicated and the program flags that signal the case an adapter will go to on a transmit request.

Case1: The first time an adapter signals a request to transmit there will be no message flags active, indicating the beginning of a message sequence. Case1 will be invoked and the adapter will reserve itself, initialize delay to 1 usec and retry to zero (reservation scheme), set M1 true, and adjust its next time to transmit by 64 usec.

Case2: This case incorporates the adapter reservation scheme (Figure 2.9). An adapter entering (Case2, Case3, Case4, Case5, Case6, or Case7) is first

checked to see if it is involved in a collision with other devices. If no collision is detected the adapter is checked for dual trunk communication, if it is, then an intermediate destination is determined. If not, then the adapter's intended receiver is checked to see if it is reserved by another adapter. If the intended receiver is not reserved, then the receiving adapter will be set reserved. Appropriate adjustments to the various transmission times, bit counts, message flags, and trunk parameters are made. If, however, the intended receiver is already reserved, the transmitting adapter will enter the retry-delay reservation scheme and its message flag will not be changed. Thus, the adapter will return to Case2 each time selected until it reserves a receiver, or aborts its transmission. If a collision is detected when entering (Case2, Case3, Case4, Case5, Case6, or Case7) then, appropriate adjustments to adapter and trunk timing is made.

Case3: This case deals with the message proper portion of the message sequence. If no collision is detected, adjustments are made to the adapter and trunk parameters.

Case4: This case is essentially the same as Case3, with the exception of the values used to update adapter and trunk parameters.

Case5: In Case5 a determination of how many data sequences need to be transmitted is made, this based on the size of the data buffer of the transmitting source. The first sequence will be updated and if more than one sequence is to be sent then message flag M4 will not be changed. Appropriate timing adjustments are made in this section also.

Case6: When all data sequences have been sent, Case6 is invoked. Appropriate timing adjustments are made in this section.

Case7: There are three possibilities encountered in this section: same trunk communication, adapter to common trunk adapter communication, and common trunk adapter to adapter communication. The last two in combination comprise a dual trunk communication. In all cases this is the termination of a message sequence. Both, the receiver and transmitter are released, and appropriate timing adjustments are made.

10 Printstats

This procedure compiles and writes to the system default output device, and auxiliary file (Auxout), an overall summary of network activity. Examples of this follow:

Single Trunk Simulation Results

```
    ***  End of Run Network Statistics ***

  Current Time :     nn.nnnn  secs

  Successful Sequence Transmissions          :nnnnn
  Collisions    (Frames)                     :nnnnn
  Waits         (Frames)                     :n.nE+nn
  Total Attempts (Sequences)                 :nnnnn
  Total Aborts                               :nnnnn

  Total Trunk Active Time          :nn.nnnn   secs
  % Total Active Time              :nn.nnn      %

  Control Bytes Transmitted        :n.nnnnnE+nn Mbytes
  Data Bytes Transmitted           :n.nnnnnE+nn Mbytes
  Total Bytes Transmitted          :n.nnnnnE+nn Mbytes

  Total Offered Load               :n.nnnnnE+nn Mbytes
```

Dual Trunk Simulation Results

```
    ***  End of Run Network Statistics ***
Current Time :  nn.nnnn   secs

Successful Sequence Transmissions          :nnnnn
Successful Sequence Transmissions-Trunk 1  :nnnnn
Successful Sequence Transmissions-Trunk 2  :nnnnn
Collisions        (Frames)                 :nnnnn
Waits             (Frames)                 :n.nE+nn
Total Attempts    (Sequences)              :nnnnn
Total Aborts                               :nnnnn
Attempted Trunk-Trunk Transmissions        :nnnnn
Successful Trunk-Trunk Transmissions      :nnnnn
Trunk 1 Active Time                        :n.nnnn secs
Trunk 2 Active Time                        :n.nnnn secs
Total Trunk Active Time                    :n.nnnn secs
% Trunk 1 Active Time                      :nn.nnn  %
% Trunk 2 Active Time                      :nn.nnn  %
% Total Trunk Active Time                  :nn.nnn  %

Control Bytes Transmitted      Trunk 1     :n.nn Mbytes
Data Bytes Transmitted         Trunk 1     :n.nn Mbytes
Control Bytes Transmitted      Trunk 2     :n.nn Mbytes
Data Bytes Transmitted         Trunk 2     :n.nn Mbytes
Total Bytes Transmitted                    :n.nn Mbytes
Total Offered Load                         :n.nn Mbytes
```

11. Activity Summary

This procedure prints detailed breakdown by device, of network activities. This summary is printed to the system default output device, and the auxiliary file (Auxout). The categories detailed are as follows:

Time Active:	Time device spent transmitting and receiving.
Time Waiting:	Time device spent waiting to transmit due to its adapter being reserved.
Time in Collisions:	Time involved in collisions by device.
Average Message Delay:	This is the average time length of a message sequence sent by this device. If this number is 0, either the device can't transmit, or it tried to but never sent an entire message sequence. In the latter case, message delay would have to be > than simulated run time.
Abort count:	Number of aborted transmission attempts (reservation scheme).
Transmission Count:	Number of successfully transmitted sequences
Reception Count:	Number of successfully received sequences
Wait Count:	Number of times device had to wait to transmit

Collision Count: Number of times engaged in a collision. (note: these numbers are actually double the total shown in overall statistics due to one count for each colliding adapter.

APPENDIX IV

Typical Single Trunk Simulation Results

```
**********************************************************
**********************************************

                NETWORK DESCRIPTION
                  ADAPTER #  1

**********************************************
**********************************************************

ADAPTER # 1:

    PRIORITY DELAY ON TRUNK 1 :  0.0000005 SEC
    FIXED DELAY:  0.0000061SEC
    TOTALDELAY ON TRUNK 1 :  0.0000190SECS
    ADAPTER RETRY COUNT :  1

    DEVICE  1   STATUS: CLOSED--DEC 11/24
                              I/O BUS TRANSFER RATE:    50000.00 BYTES/SEC
                              LOAD TIME:    0.0000025 SEC
                              NUMBER OF DATA SOURCES:  1
                              REMOTE PERIPHERAL DEVICE
                                DEVICE NUMBER  111
                                BUFFER SIZE:  2.1E+03 BYTES
                                SOURCE # 1 DATA GENERATION RATE:  15000.00 BYTES
                                TRUNK TRANSMISSION INTERVAL:  0.1397 SEC
                                DATA BLOCK COUNT :  1.02343750E+00
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:111
RECEIVER ID: 311                  PROBABILITY:  7.0E-01
RECEIVER ID: 321                  PROBABILITY:  3.0E-01

    DEVICE  2   STATUS: OPEN

    DEVICE  3   STATUS: OPEN

    DEVICE  4   STATUS: OPEN

**********************************************************
**********************************************

                NETWORK DESCRIPTION
                  ADAPTER #  2

**********************************************
**********************************************************

ADAPTER # 2:
```

```
    PRIORITY DELAY ON TRUNK 1 :  0.0000022 SEC
    FIXED DELAY:  0.0000061SEC
    TOTALDELAY ON TRUNK 1 :  0.0000190SECS
    ADAPTER RETRY COUNT :  2

    DEVICE  1   STATUS: CLOSED--VAX 3
                              I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                              LOAD TIME:    0.0000002 SEC
                              NUMBER OF DATA SOURCES:  1
                              SPACE TELESCOPE
                                DEVICE NUMBER  211
                                BUFFER SIZE:  4.0E+03 BYTES
                                SOURCE # 1 DATA GENERATION RATE:   4000.00 BYTES
                                TRUNK TRANSMISSION INTERVAL:  1.0000 SEC
                                DATA BLOCK COUNT :  1.953125000E+00
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:211
RECEIVER ID: 431                  PROBABILITY:  1.0E+00

    DEVICE  2   STATUS: CLOSED--VAX 2
                              I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                              LOAD TIME:    0.0000002 SEC
                              NUMBER OF DATA SOURCES:  1
                              SOFTWARE DEVELOPMENT
                                DEVICE NUMBER  221
                                BUFFER SIZE:  6.4E+03 BYTES
                                SOURCE # 1 DATA GENERATION RATE:   6400.00 BYTES
                                TRUNK TRANSMISSION INTERVAL:  1.0000 SEC
                                DATA BLOCK COUNT :  3.125000000E+00
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:221
RECEIVER ID: 421                  PROBABILITY:  1.0E+00

    DEVICE  3   STATUS: OPEN

    DEVICE  4   STATUS: OPEN



          ************************************************
                ***************************************

                        NETWORK DESCRIPTION
                           ADAPTER #  3

                ***************************************
          ************************************************


ADAPTER # 3:

    PRIORITY DELAY ON TRUNK 1 :  0.0000057 SEC
    FIXED DELAY:  0.0000061SEC
    TOTALDELAY ON TRUNK 1 :  0.0000190SECS
```

```
    ADAPTER RETRY COUNT :  3

    DEVICE  1   STATUS: CLOSED--VAX 4
                                I/O BUS TRANSFER RATE:    500000.00 BYTES/SEC
                                LOAD TIME:     0.0000002 SEC
                                NUMBER OF DATA SOURCES:  1
                                MPS PRIME
                                  DEVICE NUMBER  311
                                  BUFFER SIZE:  5.1E+02 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:    1000.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  0.5120 SEC
                                  DATA BLOCK COUNT :  2.500000000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:311
RECEIVER ID: 211                    PROBABILITY:  1.0E-02
RECEIVER ID: 221                    PROBABILITY:  1.0E-02

    DEVICE  2   STATUS: CLOSED--VAX 1
                                I/O BUS TRANSFER RATE:    500000.00 BYTES/SEC
                                LOAD TIME:     0.0000002 SEC
                                NUMBER OF DATA SOURCES:  1
                                MPS B/U
                                  DEVICE NUMBER  321
                                  BUFFER SIZE:  5.1E+02 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:    1000.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  0.5120 SEC
                                  DATA BLOCK COUNT :  2.500000000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:321
RECEIVER ID: 211                    PROBABILITY:  1.0E-02
RECEIVER ID: 221                    PROBABILITY:  1.0E-02

    DEVICE  3   STATUS: CLOSED--FEP
                                I/O BUS TRANSFER RATE:   3300000.00 BYTES/SEC
                                LOAD TIME:     0.0000000 SEC
                                NUMBER OF DATA SOURCES:  1
                                FEP
                                  DEVICE NUMBER  331
                                  BUFFER SIZE:  6.3E+02 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:     625.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  1.0000 SEC
                                  DATA BLOCK COUNT :  3.051757812E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:331
RECEIVER ID: 421                    PROBABILITY:  1.0E+00

    DEVICE  4   STATUS: OPEN


        ************************************************************
                ****************************************

                        NETWORK DESCRIPTION
                           ADAPTER #  4
```

C-3

```
                    ****************************************
               ***************************************************

ADAPTER # 4:

    PRIORITY DELAY ON TRUNK 1 :  0.0000083 SEC
    FIXED DELAY:  0.0000061SEC
    TOTALDELAY ON TRUNK 1 :  0.0000190SECS
    ADAPTER RETRY COUNT :  4

    DEVICE  1   STATUS: CLOSED--PE 3244 #1
                                I/O BUS TRANSFER RATE:  3300000.00 BYTES/SEC
                                LOAD TIME:    0.0000000 SEC
                                NUMBER OF DATA SOURCES:  1
                                STS PRI E
                                  DEVICE NUMBER  411
                                  BUFFER SIZE:  2.0E+03 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:      0.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999 SEC
                                  DATA BLOCK COUNT :  9.7656250000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:411

    DEVICE  2   STATUS: CLOSED--PE 8/32 # 3
                                I/O BUS TRANSFER RATE:  1200000.00 BYTES/SEC
                                LOAD TIME:    0.0000001 SEC
                                NUMBER OF DATA SOURCES:  1
                                SPACE LAB PRIME
                                  DEVICE NUMBER  421
                                  BUFFER SIZE:  2.0E+03 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:      0.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999 SEC
                                  DATA BLOCK COUNT :  9.7656250000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:421

    DEVICE  3   STATUS: CLOSED--PE 8/32 # 4
                                I/O BUS TRANSFER RATE:  1200000.00 BYTES/SEC
                                LOAD TIME:    0.0000001 SEC
                                NUMBER OF DATA SOURCES:  1
                                SPACE LAB B/U
                                  DEVICE NUMBER  431
                                  BUFFER SIZE:  2.0E+03 BYTES
                                  SOURCE # 1 DATA GENERATION RATE:      0.00 BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999 SEC
                                  DATA BLOCK COUNT :  9.7656250000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:431

    DEVICE  4   STATUS: OPEN
```

*** END OF RUN NETWORK STATISTICS ***

CURRENT TIME : 40.0508 SECS

SUCCESSFUL SEQUENCE TRANSMISSIONS	:	556
COLLISIONS (FRAMES)	:	1
WAITS (FRAMES)	:	6.7E+01
TOTAL ATTEMPTS (SEQUENCES)	:	557
TOTAL ABORTS	:	0
TOTAL TRUNK ACTIVE TIME	:	0.4781966 SECS
% TOTAL ACTIVE TIME	:	1.194 %
CONTROL BYTES TRANSMITTED	:	1.04252E+00 MBYTES
DATA BYTES TRANSMITTED	:	1.87793E+00 MBYTES
TOTAL BYTES TRANSMITTED	:	2.92045E+00 MBYTES
TOTAL OFFERED LOAD	:	1.12242E+00 MBYTES

DEVICE ACTIVITY SUMMARIES
(SECONDS)

ADP #	DEV #	SOURCE #	TIME ACTIVE	TIME WAITING	TIME IN COLLISIONS	AVG MESSAGE DELAY	ABORT COUNT	TRANSMISSION COUNT	RECEPTION COUNT	WAIT COUNT	COLLISION COUNT
1	1	1	0.2778	0.0000	0.0E+00	1.3E-03	0	283	0	0	0
REMOTE PERIPHERAL DEVICE											
2	1	1	0.1010	0.0000	1.2E-06	1.7E-03	0	39	156	0	1
SPACE TELESCOPE											
2	2	1	0.0736	0.0017	0.0E+00	2.4E-03	0	39	0	38	0
SOFTWARE DEVELOPMENT											
3	1	1	0.3086	0.0000	0.0E+00	5.3E-04	0	78	283	0	0
MPS PRIME											
3	2	1	0.0232	0.0005	0.0E+00	5.3E-04	0	78	0	11	0

MPS B/U

3 3 1 FEP	0.0151	0.0011	1.2E-06	1.7E-03	0	39	0	18	1
4 1 1 STS PRIME	0.0000	0.0000	0.0E+00	0.0E+00	0	0	0	0	0
4 2 1 SPACE LAB PRIME	0.0908	0.0000	0.0E+00	0.0E+00	0	0	78	0	0
4 3 1 SPACE LAB B/U	0.0515	0.0000	0.0E+00	0.0E+00	0	0	39	0	0

APPENDIX V

Typical Dual Trunk Simulation Results

```
          ************************************************************
                  ****************************************

                              NETWORK DESCRIPTION
                                 ADAPTER #  1

                  ****************************************
          ************************************************************


ADAPTER # 1:

    PRIORITY DELAY ON TRUNK 1 :  0.0000005 SEC
    FIXED DELAY:  0.0000041SEC
    TOTALDELAY ON TRUNK 1 :  0.0000118SECS
    ADAPTER RETRY COUNT :  1

    DEVICE  1   STATUS: CLOSED--PE 3244 CSO
                              I/O BUS TRANSFER RATE:  3300000.00 BYTES/SEC
                              LOAD TIME:    0.0000000 SEC
                              NUMBER OF DATA SOURCES:  1
                              DATA GENERATOR
                                DEVICE NUMBER  111
                                BUFFER SIZE:  1.0E+03 BYTES
                                SOURCE # 1 DATA GENERATION RATE:  10000.00 BYTES
                                TRUNK TRANSMISSION INTERVAL:  0.1024 SEC
                                DATA BLOCK COUNT :  5.0000000E-01
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:111
RECEIVER ID: 521                  PROBABILITY:  1.0E+00

    DEVICE  2   STATUS: OPEN

    DEVICE  3   STATUS: OPEN

    DEVICE  4   STATUS: OPEN



          ************************************************************
                  ****************************************

                              NETWORK DESCRIPTION
                                 ADAPTER #  2

                  ****************************************
          ************************************************************


ADAPTER # 2:

    PRIORITY DELAY ON TRUNK 1 :  0.0000041 SEC
```

```
    FIXED DELAY:  0.0000041SEC
    TOTALDELAY ON TRUNK 1 :  0.0000118SECS
    ADAPTER RETRY COUNT :  2

    PRIORITY DELAY ON TRUNK 2 :  0.0000C5SEC
    FIXED DELAY:  0.0000061SEC
    TOTAL DELAY ON TRUNK 2 :  0.0000190SEC

    DEVICE  1   STATUS: CLOSED--DEC 11/24
                              I/O BUS TRANSFER RATE:    50000.00 BYTES/SEC
                              LOAD TIME:    0.0000025 SEC
                              NUMBER OF DATA SOURCES:  1
                              REMOTE PERIPHERAL DEVICE
                                DEVICE NUMBER  211
                                BUFFER SIZE:  2.1E+03 BYTES
                                SOURCE #.1 DATA GENERATION RATE:  15000.00 BYTES
                                TRUNK TRANSMISSION INTERVAL:  0.1397 SEC
                                DATA BLOCK COUNT :  1.023437500E+00
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR:211
RECEIVER ID: 411                  PROBABILITY:  7.0E-01
RECEIVER ID: 421                  PROBABILITY:  3.0E-01

    DEVICE  2   STATUS: OPEN

    DEVICE  3   STATUS: OPEN

    DEVICE  4   STATUS: OPEN
```



```
      **********************************************************
            ****************************************

                        NETWORK DESCRIPTION
                           ADAPTER #   3

            ****************************************
      **********************************************************

ADAPTER #  3:

    PRIORITY DELAY ON TRUNK 2 :  0.0000022SECS
    FIXED DELAY ON TRUNK 2:  0.0000061SECS
    TOTAL DELAY ON TRUNK 2:  0.0000190SECS
    ADAPTER RETRY COUNT :  3

    DEVICE  1  STATUS : CLOSED--VAX 3
                              I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                              LOAD TIME:    0.0000002 SEC
                              NUMBER OF DATA SOURCES:  1
                              SPACE TELESCOPE
```

```
                                DEVICE NUMBER  311
                                BUFFER SIZE:  4.0E+03BYTES
                                SOURCE # 1DATA GENERATION RATE:   4000.00BYTES
                                TRUNK TRANSMISSION INTERVAL:  1.0000SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 311
RECEIVER ID: 531                  PROBABILITY:  1.0E+00

    DEVICE  2  STATUS : CLOSED--VAX 2
                               I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                               LOAD TIME:    0.0000002 SEC
                               NUMBER OF DATA SOURCES:  1
                               SOFTWARE DEVELOPMENT
                                DEVICE NUMBER  321
                                BUFFER SIZE:  6.4E+03BYTES
                                SOURCE # 1DATA GENERATION RATE:   6400.00BYTES
                                TRUNK TRANSMISSION INTERVAL:  1.0000SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 321
RECEIVER ID: 521                  PROBABILITY:  1.0E+00
RECEIVER ID: 111                  PROBABILITY:  3.0E-01

    DEVICE  3  STATUS : OPEN

    DEVICE  4  STATUS : OPEN

          ******************************************************
                ****************************************

                            NETWORK DESCRIPTION
                               ADAPTER #   4

                ****************************************
          ******************************************************


ADAPTER #  4:

    PRIORITY DELAY ON TRUNK 2 :  0.0000057SECS
    FIXED DELAY ON TRUNK 2:  0.0000061SECS
    TOTAL DELAY ON TRUNK 2:  0.0000190SECS
    ADAPTER RETRY COUNT :   4

    DEVICE  1  STATUS : CLOSED--VAX 4
                               I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                               LOAD TIME:    0.0000002 SEC
                               NUMBER OF DATA SOURCES:  1
                               MPS PRIME
                                DEVICE NUMBER  411
                                BUFFER SIZE:  5.1E+02BYTES
                                SOURCE # 1DATA GENERATION RATE:   1000.00BYTES
                                TRUNK TRANSMISSION INTERVAL:  0.5120SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 411
RECEIVER ID: 311                  PROBABILITY:  1.0E-02
RECEIVER ID: 321                  PROBABILITY:  1.0E-02

    DEVICE  2  STATUS : CLOSED--VAX 1
```

```
                                I/O BUS TRANSFER RATE:   500000.00 BYTES/SEC
                                LOAD TIME:     0.0000002 SEC
                                NUMBER OF DATA SOURCES:  1
                                MPS B/U
                                  DEVICE NUMBER  421
                                  BUFFER SIZE:  5.1E+02BYTES
                                  SOURCE # 1DATA GENERATION RATE:   1000.00BYTES
                                  TRUNK TRANSMISSION INTERVAL:  0.5120SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 421
RECEIVER ID: 311                    PROBABILITY:  1.0E-02
RECEIVER ID: 321                    PROBABILITY:  1.0E-02

    DEVICE  3  STATUS : CLOSED--FEP
                                I/O BUS TRANSFER RATE:  3300000.00 BYTES/SEC
                                LOAD TIME:     0.0000000 SEC
                                NUMBER OF DATA SOURCES:  1
                                FEP
                                  DEVICE NUMBER  431
                                  BUFFER SIZE:  6.3E+02BYTES
                                  SOURCE # 1DATA GENERATION RATE:    625.00BYTES
                                  TRUNK TRANSMISSION INTERVAL:  1.0000SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 431
RECEIVER ID: 521                    PROBABILITY:  1.0E+00

    DEVICE  4  STATUS : OPEN

          ********************************************************
                ****************************************

                            NETWORK DESCRIPTION
                              ADAPTER #   5

                ****************************************
          ********************************************************


ADAPTER #  5:

    PRIORITY DELAY ON TRUNK 2 :  0.0000083SECS
    FIXED DELAY ON TRUNK 2:  0.0000061SECS
    TOTAL DELAY ON TRUNK 2:  0.0000190SECS
    ADAPTER RETRY COUNT :   5

    DEVICE  1  STATUS : CLOSED--PE 3244 #1
                                I/O BUS TRANSFER RATE:  3300000.00 BYTES/SEC
                                LOAD TIME:     0.0000000 SEC
                                NUMBER OF DATA SOURCES:  1
                                STS PRIME
                                  DEVICE NUMBER  511
                                  BUFFER SIZE:  2.0E+03BYTES
                                  SOURCE # 1DATA GENERATION RATE:      0.00BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 511

    DEVICE  2  STATUS : CLOSED--PE 9 32 # 3
```

```
                                I/O BUS TRANSFER RATE:  1200000.00 BYTES/SEC
                                LOAD TIME:     0.0000001 SEC
                                NUMBER OF DATA SOURCES:  1
                                SPACE LAB PRIME
                                  DEVICE NUMBER  521
                                  BUFFER SIZE:  2.0E+03BYTES
                                  SOURCE # 1DATA GENERATION RATE:       0.00BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 521

    DEVICE  3  STATUS : CLOSED--PE 8/32 # 4
                                I/O BUS TRANSFER RATE:  1200000.00 BYTES/SEC
                                LOAD TIME:     0.0000001 SEC
                                NUMBER OF DATA SOURCES:  1
                                SPACE LAB B/U
                                  DEVICE NUMBER  531
                                  BUFFER SIZE:  2.0E+03BYTES
                                  SOURCE # 1DATA GENERATION RATE:       0.00BYTES
                                  TRUNK TRANSMISSION INTERVAL:  9999999.9999SECS
POSSIBLE RECEIVERS AND THEIR RESPECTIVE PROBABILITIES FOR: 531

    DEVICE  4  STATUS : OPEN
```



```
          *** END OF RUN NETWORK STATISTICS ***

CURRENT TIME :       40.0651 SECS

SUCCESSFUL SEQUENCE TRANSMISSIONS          :  1049
SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK 1  :   388
SUCCESSFUL SEQUENCE TRANSMISSIONS-TRUNK 2  :   661
COLLISIONS    (FRAMES)                     :     1
WAITS         (FRAMES)                     :  4.2E+01
TOTAL ATTEMPTS (SEQUENCES)                 :  1050
TOTAL ABORTS                               :     0
ATTEMPTED TRUNK-TRUNK TRANSMISSIONS        :   388
SUCCESSFUL TRUNK-TRUNK TRANSMISSIONS       :   388
```

TRUNK 1 ACTIVE TIME	:	0.1579856 SECS
TRUNK 2 ACTIVE TIME	:	0.3538636 SECS
TOTAL TRUNK ACTIVE TIME	:	0.5118492 SECS
% TRUNK 1 ACTIVE TIME	:	0.394 %
% TRUNK 2 ACTIVE TIME	:	0.893 %
% TOTAL ACTIVE TIME	:	1.278 %
CONTROL BYTES TRANSMITTED - TRUNK 1	:	5.30057E-01 MBYTES
DATA BYTES TRANSMITTED - TRUNK 1	:	4.22144E-01 MBYTES
CONTROL BYTES TRANSMITTED - TRUNK 2	:	1.01627E+00 MBYTES
DATA BYTES TRANSMITTED - TRUNK 2	:	1.10921E+00 MBYTES
TOTAL BYTES TRANSMITTED	:	3.07768E+00 MBYTES
TOTAL OFFERED LOAD	:	1.52347E+00 MBYTES

DEVICE ACTIVITY SUMMARIES
(SECONDS)

ADP #	DEV #	SOURCE #	TIME ACTIVE	TIME WAITING	TIME IN COLLISIONS	AVG MESSAGE DELAY	ABORT COUNT	TRANSMISSION COUNT	RECEPTION COUNT	WAIT COUNT	COLLISION COUNT
1	1	1	0.1476	0.0000	0.0E+00	6.6E-04	0	388	0	0	0
DATA GENERATOR											
2	1	1	0.3081	0.0000	0.0E+00	6.7E-04	0	388	388	0	0
REMOTE PERIPHERAL DEVICE											
3	1	1	0.1010	0.0000	1.2E-06	1.7E-03	0	39	156	0	1
SPACE TELESCOPE											
3	2	1	0.0736	0.0017	0.0E+00	2.4E-03	0	39	0	31	0
SOFTWARE DEVELOPMENT											
4	1	1	0.0232	0.0000	0.0E+00	5.8E-04	0	78	0	0	0
MPS PRIME											
4	2	1	0.0232	0.0005	0.0E+00	5.8E-04	0	78	0	11	0
MPS B/U											
4	3	1	0.0152	0.0000	1.2E-06	1.7E-03	0	39	0	0	1
FEP											
5	1	1	0.0000	0.0000	0.0E+00	0.0E+00	0	0	0	0	0
STS PRIME											
5	2	1	0.2518	0.0000	0.0E+00	0.0E+00	0	0	466	0	0
SPACE LAB PRIME											
5	3	1	0.0515	0.0000	0.0E+00	0.0E+00	0	0	39	0	0
SPACE LAB B/U											

REFERENCES AND BIBLIOGRAPHY

DEC80 Digital Equipment Corporation, VAX Hardware Handbook. Maynard, Ma: 1981.

DEC81 Digital Equipment Corporation, VAX Architecture. Maynard, Ma: 1981

DEC82 Digital Equipment Corporation, VAX Software Handbook, Maynard, Ma:1981-1982.

DONN79 Donnelly, J. E., and Yeh, J. W. "Interaction between Protocol Levels in a Prioritized CSMA Broadcast Network." Computer Networks, March 79

FRAN80 Franta, W. R., and Bilodeau, M. B. "Analysis of a Prioritized CSMA Protocol Based on Staggered Delays." Acta Informatica, June, 1980.

FRAN82 Franta, W. R., and Heath, M. B. "Performance of HYPERchannel Networks: Parameters, Measurements, Models, and Analysis." University of Minnesota, Computer Science Department, Technical Report 82-3, January, 1982.

FRAN84 Franta, W. R., and Heath, M. B. "Measurement and Analsis of HYPERchannel Networks." IEEE Transactions on Computers, March, 1984.

HEYM82 Heyman, D. P. "An Analysis of the Carrier-Sense Multiple Access Protocol." The Bell System Technical Journal, Vol.61, No.8, Oct., 1982.

MAUL84 Mauldin, J. D. "A Software Model of the HYPERchannel Local Area Network at the Huntsville Operation Support Center." Thesis submitted to Mississippi State University, Mississippi State, Ms. 1984.

METC76 Metcalfe, R. M., and Boffs, D. R. "Ethernet: Packet Switching for Local Networks." Communications of the ACM, July, 1976.

NASA82 National Aeronautics and Space Administration, George C. Marshall Space Flight Center. "Shuttle Program: HOSC Operations Procedures Document." SPMD 1.2.1, Revision 6, May, 1982.

NSC79 Network Systems Corporation. A400 Adapter Reference Manual, 42990008. Brooklyn Park, MN: 1979.

NSC80a Network Systems Corporation. Nucleus Adapter Reference Manual, 4290003. Brooklyn Park, MN: 1980.

NSC80b Network Systems Corporation. HYPERchannel Systems Description, A01-000002. Brooklyn Park, MN: 1980.

NSC81 Network System Corporation. PI 13/14 Peripheral Interface Reference Manual, 42990015. Brooklyn Park, MN: 1981.

PEC81 Perkin Elmer Corporation. "Model 3240 Processors," Product Information Sheet. Ocean Port, NJ: 1981.

STAL84 Stallings, W. Local Networks. New York: MacMillan Publishing Company, 1984.

WATS82 Watson, B. W. "Validation of a Discrete Event Computer Model of Network Systems Corporation's HYPERchannel". 7th Conference on Local Computer Networks. Minneapolis, MN: 1982.

YEH79 Yeh, J. W. "Simulations of Local Computer Networks". 4th Conference on Local Computer Networks. Minneapolis, MN: 1979.